

TIA Portal Openness: Automating creation of projects

System Manual

Online help printout

Security note

1

Readme TIA Portal
Openness

2

What's new in TIA Portal
Openness?

3

Basics

4

Introduction

5

Configurations

6

TIA Portal Openness API

7

Export/import

8

Major Changes

9

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

DANGER

indicates that death or severe personal injury **will** result if proper precautions are not taken.

WARNING

indicates that death or severe personal injury **may** result if proper precautions are not taken.

CAUTION

indicates that minor personal injury can result if proper precautions are not taken.

NOTICE

indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

WARNING

Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Table of contents

1	Security note.....	17
2	Readme TIA Portal Openness.....	19
2.1	Readme.....	19
2.2	Major changes in TIA Portal Openness V16	22
2.3	Announcement of major changes in future releases.....	24
2.4	Hints for writing long-term stable code.....	25
3	What's new in TIA Portal Openness?.....	27
4	Basics	29
4.1	Requirements for TIA Portal Openness	29
4.2	Installation	30
4.2.1	Installing TIA Portal Openness.....	30
4.2.2	Adding users to the "Siemens TIA Openness" user group	31
4.2.3	Accessing the TIA Portal.....	36
4.3	Openness tasks	37
4.3.1	Applications.....	37
4.3.2	Export/import.....	38
4.4	Object list	38
4.5	Setting attributes of Blocks, DBs & UDTs	42
4.6	Standard libraries	43
4.7	Notes on performance of TIA Portal Openness	44
5	Introduction.....	45
6	Configurations	47
7	TIA Portal Openness API	51
7.1	Introduction	51
7.2	Programming steps	51
7.3	TIA Portal Openness object model	52
7.4	Blocks and types of the TIA Portal Openness object model	58
7.5	Hierarchy of hardware objects of the object model	66
7.6	Information about installed TIA Portal Openness versions	68
7.7	Application example: Creating API access in a Windows Forms application.....	69
7.8	Use of the code examples.....	74
7.9	General functions	76
7.9.1	TIA Portal Openness IntelliSense support	76

7.9.2	Connecting to the TIA Portal	76
7.9.3	TIA Portal Openness firewall.....	81
7.9.4	Event handlers	82
7.9.5	Program-controlled acknowledgement of dialogs with system events.....	84
7.9.6	Terminating the connection to the TIA Portal.....	85
7.9.7	Diagnostic interfaces on TIA Portal.....	86
7.9.8	Exclusive access.....	91
7.9.9	Dynamic behaviours.....	93
7.9.10	Working with associations.....	96
7.9.11	Working with compositions.....	96
7.9.12	Verifying object equality	97
7.9.13	Read operations for attributes	98
7.9.14	Transaction handling	100
7.9.15	Creating a DirectoryInfo/FileInfo object.....	102
7.9.16	Self-description support for attributes, navigators, actions, and services	103
7.10	Functions for projects and project data	106
7.10.1	Opening a project.....	106
7.10.2	Creating a project.....	110
7.10.3	Accessing general settings of the TIA Portal	111
7.10.4	Archiving and Retrieving a project	115
7.10.5	Accessing read-only TIA Portal project	118
7.10.6	Accessing languages	119
7.10.7	Determining the object structure and attributes	121
7.10.8	Access software target	123
7.10.9	Accessing and enumerating multilingual texts	124
7.10.10	Updating project property simulation support.....	125
7.10.11	Read project related attributes	126
7.10.12	Deleting project graphics.....	129
7.10.13	Compiling a project	129
7.10.14	Saving a project	131
7.10.15	Closing a project	133
7.11	Functions for Connections	133
7.11.1	Configurable attributes of a port-to-port connection.....	133
7.12	Functions on libraries	136
7.12.1	Functions for objects and instances.....	136
7.12.2	Accessing global libraries.....	137
7.12.3	Accessing global library languages	139
7.12.4	Opening libraries	141
7.12.5	Enumerating open libraries	142
7.12.6	Saving and closing libraries	143
7.12.7	Archiving and retrieving a library	145
7.12.8	Creating global libraries	147
7.12.9	Accessing folders in a library	148
7.12.10	Accessing types	151
7.12.11	Accessing type versions.....	154
7.12.12	Accessing blocks located in library	158
7.12.13	Accessing instances	160
7.12.14	Accessing master copies	162
7.12.15	Create master copy from a project in library	165
7.12.16	Create an object from a master copy	166
7.12.17	Copying master copies.....	168

7.12.18	Determining out-of-date type instances	169
7.12.19	Updating the project	172
7.12.20	Updating a library	174
7.12.21	Deleting library content	175
7.13	Functions for accessing devices, networks and connections.....	177
7.13.1	Open the "Devices & networks" editor	177
7.13.2	Querying PLC and HMI targets	178
7.13.3	Accessing attributes of an address object.....	180
7.13.4	Accessing the channels of a module.....	182
7.14	Functions on networks	184
7.14.1	Creating a subnet.....	184
7.14.2	Accessing subnets	185
7.14.3	Accessing internal subnets	186
7.14.4	Get type identifier of subnets	187
7.14.5	Accessing attributes of a subnet	188
7.14.6	Deleting a global subnet.....	194
7.14.7	Enumerate all participants of a subnet.....	194
7.14.8	Enumerate IO systems of a subnet.....	195
7.14.9	Accessing nodes	195
7.14.10	Accessing attributes of a node	196
7.14.11	Connecting a node to a subnet	200
7.14.12	Disconnect a node from a subnet	200
7.14.13	Creating an IO system	201
7.14.14	Accessing the attributes of an IO system.....	202
7.14.15	Connecting an IO connector to an IO system	202
7.14.16	Get master system or IO system of an interface.....	203
7.14.17	Get an IO Controller	204
7.14.18	Get an IO Connector	205
7.14.19	Disconnecting an IO connector from an IO system or a DP mastersystem	206
7.14.20	Accessing attributes of a DP mastersystem.....	206
7.14.21	Accessing attributes of a profinet io system.....	207
7.14.22	Deleting a DP master system.....	209
7.14.23	Deleting a profinet io system.....	209
7.14.24	Creating a DP master system	210
7.14.25	Accessing port interconnection information of port device item	211
7.14.26	Attributes of port inter-connection	212
7.14.27	Accessing the attributes of a port.....	215
7.14.28	Enumerate DP master systems of a subnet.....	216
7.14.29	Enumerate assigned IO connectors.....	216
7.14.30	Connecting a DP IO connector to a DP master system	217
7.14.31	Accessing AS-i profile and parameter attributes for virtual slaves	218
7.15	Functions on devices	220
7.15.1	Mandatory attributes of devices	220
7.15.2	Get type identifier of devices and device items.....	221
7.15.3	Set App ID on device and device Items	224
7.15.4	Get App ID on device and device Items	225
7.15.5	Remove App ID on device and device items	226
7.15.6	Creating a device	227
7.15.7	Enumerating devices.....	228
7.15.8	Accessing devices.....	231
7.15.9	Deleting a device.....	233

7.16	Functions on device items.....	234
7.16.1	Mandatory attributes of device items	234
7.16.2	Creating and plugging a device item.....	236
7.16.3	Moving device items into another slot.....	239
7.16.4	Copying a device item.....	240
7.16.5	Deleting a device item.....	241
7.16.6	Enumerate device items	242
7.16.7	Accessing device items.....	243
7.16.8	Accessing device item as interface	247
7.16.9	Accessing attributes of an I/O device interface	248
7.16.10	Accessing attributes of IoController	250
7.16.11	Accessing attributes of IoConnector	251
7.16.12	Accessing address controller	253
7.16.13	Accessing addresses	254
7.16.14	Accessing hardware identifiers	256
7.16.15	Accessing hardware identifier controller	257
7.16.16	Accessing channels of device items	258
7.16.17	Creating and exporting psc file.....	259
7.16.18	Connection handling for extension racks	260
7.17	Functions for accessing the data of an HMI device	261
7.17.1	Screens	261
7.17.1.1	Creating user-defined screen folders	261
7.17.1.2	Deleting a screen from a folder	262
7.17.1.3	Deleting a screen template from a folder	262
7.17.1.4	Deleting all screens from a folder.....	263
7.17.2	Cycles	264
7.17.2.1	Deleting a cycle.....	264
7.17.3	Text lists	265
7.17.3.1	Deleting a text list.....	265
7.17.4	Graphic lists	266
7.17.4.1	Deleting a graphic list.....	266
7.17.5	Connections	266
7.17.5.1	Deleting a connection.....	266
7.17.6	Tag table	267
7.17.6.1	Creating user-defined folders for HMI tags	267
7.17.6.2	Enumerating tags of an HMI tag table.....	267
7.17.6.3	Deleting an individual tag from an HMI tag table	268
7.17.6.4	Deleting a tag table from a folder	269
7.17.7	VB scripts	269
7.17.7.1	Creating user-defined script folders	269
7.17.7.2	Deleting a VB script from a folder	270
7.17.8	Deleting a user-defined folder of an HMI device	271
7.18	Functions for accessing the data of an HMI device (Unified).....	271
7.18.1	Object list	271
7.18.2	HMI Unified Software object.....	272
7.18.3	Querying errors	274
7.18.4	Alarms	277
7.18.4.1	Working with analog alarms	277
7.18.4.2	Working with discrete alarms	280
7.18.4.3	Working with alarm classes.....	283
7.18.5	Logs	286

7.18.5.1	Working with data logs	286
7.18.5.2	Working with alarm logs	289
7.18.5.3	Working with logging tags	292
7.18.6	Tags and tag tables.....	296
7.18.6.1	Working with tag tables	296
7.18.6.2	Working with HMI tags	297
7.18.6.3	Working with system tags	306
7.18.7	Connections	307
7.18.7.1	Working with connections	307
7.18.8	Runtime settings	311
7.18.8.1	Working with runtime settings	311
7.18.9	Screens and dynamizations	312
7.18.9.1	Working with screens	312
7.18.9.2	Basic screen items	317
7.18.9.3	Element screen items.....	340
7.18.9.4	Control screen items	369
7.18.9.5	Faceplate instance screen items.....	390
7.18.9.6	Working with events for screen/screen items & properties	398
7.18.9.7	Working with dynamization & events for screen/screen items using scripts	401
7.18.9.8	Working with dynamization for screens/screen items	404
7.18.9.9	Checking license for access Unified device	407
7.18.10	Accessing common plant model hierarchies.....	410
7.18.10.1	Working with plant view.....	410
7.18.10.2	Working with plant view nodes.....	411
7.18.10.3	Enumerating plant view	412
7.18.10.4	Working with plant view to device	414
7.18.11	Accessing common plant model instance	415
7.18.11.1	Working with CPM object instance	415
7.18.11.2	Working with plant view nodes of CPM object instance	417
7.18.11.3	Enumerating interface tags of CPM object instance	418
7.18.12	Accessing properties for interface/logging tags of plant object instances	419
7.18.12.1	Accessing/Updating properties of interface tags of CPM plant object instances	419
7.18.12.2	Accessing/Updating properties of members tags of interface tags	421
7.18.12.3	Accessing/Updating properties of logging tags of member tags	423
7.19	Functions for accessing the data of a PLC device	425
7.19.1	Determining the status of a PLC	425
7.19.2	Accessing parameters of an online connection.....	426
7.19.3	Accessing fingerprint for quick station compare	430
7.19.4	Accessing CM DP as DP slave and transfer area.....	438
7.19.5	Setting PLC online of R/H system.....	439
7.19.6	Accessing software container from primary PLC of R/H system	441
7.19.7	Downloading PLCs of R/H System	442
7.19.8	Functions for downloading data to PLC device.....	448
7.19.8.1	Downloading PC System	448
7.19.8.2	Downloading hardware and software components to PLC device	450
7.19.8.3	Running and stopping PLC	460
7.19.8.4	Supporting callbacks	461
7.19.8.5	Protecting PLC through password	463
7.19.8.6	Handling PLC block binding passwords	464
7.19.9	Functions for accessing PLC service	465
7.19.9.1	Access level setting.....	465
7.19.9.2	Accessing OPC UA server interface	467

7.19.9.3	Accessing Software Checksum.....	469
7.19.9.4	Assigning PC interface	470
7.19.10	Uploading hardware, software and files to PLC device.....	472
7.19.11	Comparing PLC software	478
7.19.12	Comparing PLC hardware.....	481
7.19.13	Establishing or disconnecting the online connection to the PLC.....	482
7.19.14	Assigning project language to PLC	483
7.19.15	Assigning watch & force tables for web server and PLC display	484
7.19.16	Accessing web server and OPC UA user management	487
7.19.17	Accessing domain settings.....	490
7.19.18	Setting a display password	491
7.19.19	Supporting IP Accessibility.....	492
7.19.20	Updating module description.....	493
7.19.21	Managing certificate	494
7.19.22	Blocks.....	498
7.19.22.1	Querying the "Program blocks" group	498
7.19.22.2	Querying the system group for system blocks	498
7.19.22.3	Enumerating system subgroups.....	499
7.19.22.4	Enumerating user-defined block groups	500
7.19.22.5	Enumerating all blocks	501
7.19.22.6	Querying information of a block/user data type.....	502
7.19.22.7	Setting and removing protections from a block	504
7.19.22.8	Deleting block.....	506
7.19.22.9	Creating group for blocks	507
7.19.22.10	Deleting group for blocks	508
7.19.22.11	Accessing attributes of all blocks	508
7.19.22.12	Creating a ProDiag-FB	510
7.19.22.13	Accessing supervisions and properties of ProDiag-FB	511
7.19.22.14	Reading ProDiag-FB blocks and attributes	513
7.19.22.15	Adding an external file.....	513
7.19.22.16	Generate source from block.....	514
7.19.22.17	Generating blocks from source	516
7.19.22.18	Generating from source of known source format	517
7.19.22.19	Deleting user data type	520
7.19.22.20	Deleting an external file.....	520
7.19.22.21	Starting the block editor	521
7.19.22.22	Changing blocks using fingerprints	522
7.19.22.23	Generating/deleting blocks for user defined pages	523
7.19.23	Technology objects	525
7.19.23.1	Overview of functions for technology objects	525
7.19.23.2	Overview of technology objects and versions	526
7.19.23.3	Overview of data types.....	528
7.19.23.4	Querying the composition of technology objects.....	529
7.19.23.5	Creating technology object.....	529
7.19.23.6	Deleting technology object	530
7.19.23.7	Compiling technology object	531
7.19.23.8	Enumerating technology object.....	532
7.19.23.9	Finding technology object	533
7.19.23.10	Enumerating parameters of technology object	534
7.19.23.11	Finding parameters of technology object	534
7.19.23.12	Reading parameters of technology object.....	535
7.19.23.13	Writing parameters of technology object.....	536

7.19.23.14	S7-1200 Motion Control	537
7.19.23.15	S7-1500 Motion Control	545
7.19.23.16	PID control	563
7.19.23.17	Counting	564
7.19.23.18	Easy Motion Control	564
7.19.24	Tags and Tag tables	565
7.19.24.1	Starting the "PLC Tags" editor	565
7.19.24.2	Querying system groups for PLC tags	566
7.19.24.3	Creating PLC tag table	566
7.19.24.4	Enumerating user-defined groups for PLC tags	567
7.19.24.5	Creating user-defined groups for PLC tags	568
7.19.24.6	Deleting user-defined groups for PLC tags	569
7.19.24.7	Enumerating PLC tag tables in a folder	569
7.19.24.8	Querying information from a PLC tag table	570
7.19.24.9	Reading the time of the last changes of a PLC tag table	572
7.19.24.10	Deleting a PLC tag table from a group	572
7.19.24.11	Enumerating PLC tags	573
7.19.24.12	Accessing PLC tags	573
7.19.24.13	Accessing PLC constants	575
7.19.25	Functions for software units	577
7.19.25.1	Working with software unit	577
7.19.25.2	Accessing software unit	580
7.19.25.3	Accessing software unit underlying objects	581
7.19.25.4	Accessing to existing relations of a unit	583
7.19.25.5	Updating software unit properties	585
7.19.25.6	Publishing software unit object	587
7.19.25.7	Adding external sources in units	589
7.19.25.8	Units as mastercopies	591
7.19.25.9	Updating existing relations & create/delete relations	593
7.20	Functions for Version Control Interface	599
7.20.1	Accessing VCI system group in project	599
7.20.2	Enumerating user groups in VCI group	599
7.20.3	Creating VCI user group in VCI group	600
7.20.4	Updating VCI group properties	601
7.20.5	Deleting VCI user group	602
7.20.6	Enumerating VCI workspaces in VCI group	602
7.20.7	Creating VCI workspace in VCI group	603
7.20.8	Updating VCI workspace properties	604
7.20.9	Deleting VCI workspace	605
7.20.10	Enumerating VCI workspace mappings in VCI	606
7.20.11	Creating VCI workspace mapping in VCI workspace	607
7.20.12	Updating VCI workspace mapping properties	608
7.20.13	Deleting VCI workspace mapping	608
7.20.14	Synchronizing a workspace mapping	609
7.21	Functions on OPC	612
7.21.1	Configuring OPC UA server secure communication protocol	612
7.21.2	Setting OPC UA security policy	614
7.22	SiVArc Openness	615
7.22.1	Introduction	615
7.22.2	SiVArc service properties	616
7.22.3	Copying rules or groups from library	617

7.22.4	Finding a screen rule and screen rule group.....	619
7.22.5	Deleting rules and rule groups	620
7.22.6	UMAC set up for openness	621
7.22.7	SiVArc generation	621
7.23	Openness for CP 1604/CP 1616/CP 1626.....	623
7.24	Openness transfer areas for PnPn coupler.....	627
7.25	Openness virtual modules/submodules for ET 200SP PN HF	630
7.26	Functions for SINUMERIK	632
7.26.1	Introduction	632
7.26.2	Type identifier - identifier of the components	633
7.26.3	Fundamentals	633
7.26.4	Object model	637
7.26.5	Reference.....	639
7.26.5.1	DriveObject	639
7.26.5.2	DriveObjectContainer.....	639
7.26.5.3	DriveObjectComposition	640
7.26.5.4	AddressComposition	640
7.26.5.5	AddressContext.....	642
7.26.5.6	AddressIoType	642
7.26.6	Code example	642
7.26.6.1	General	642
7.26.6.2	Executing the first steps in SINUMERIK	643
7.26.6.3	Creating an NCU.....	643
7.26.6.4	Creating an NX module.....	643
7.26.6.5	Connecting an NX module with NCU	644
7.26.6.6	Creating archives	645
7.26.6.7	Activating Safety Integrated	647
7.27	Functions for SINUMERIK ONE.....	648
7.27.1	Introduction	648
7.27.2	Type identifier designation of the components.....	648
7.27.3	Fundamentals	649
7.27.4	Object model	652
7.27.5	Reference.....	654
7.27.5.1	Namespace Siemens.Engineering.MC.DriveConfiguration	654
7.27.5.2	Namespace Siemens.Engineering.MC.Drives	660
7.27.5.3	ArchiveProvider	669
7.27.6	Code example	669
7.27.6.1	General	669
7.27.6.2	Executing the first steps in SINUMERIK	669
7.27.6.3	Creating an NCU.....	670
7.27.6.4	Creating an NX module.....	670
7.27.6.5	Connecting an NX module with NCU	671
7.27.6.6	Accessing NCK events.....	671
7.27.6.7	Creating archives	672
7.27.6.8	Activating Safety Integrated	674
7.27.6.9	Examples for Namespace Siemens.Engineering.MC.DriveConfiguration	675
7.27.6.10	Examples for Namespace Siemens.Engineering.MC.Drives	679
7.28	Functions for Startdrive	683
7.28.1	Introduction	683

7.28.2	TypeIdentifier - identifier of the components	684
7.28.3	References	685
7.28.3.1	AddressComposition	685
7.28.3.2	AddressContext	686
7.28.3.3	AddressIoType	686
7.28.3.4	ConfigurationEntry	687
7.28.3.5	DriveDomainFunctions	687
7.28.3.6	DriveObject	688
7.28.3.7	DriveObjectActivation	688
7.28.3.8	DriveObjectContainer	689
7.28.3.9	DriveObjectTypeHandler	689
7.28.3.10	DriveParameter	690
7.28.3.11	DriveParameterComposition	691
7.28.3.12	EncoderConfiguration	692
7.28.3.13	HardwareProjection	693
7.28.3.14	MotorConfiguration	694
7.28.3.15	OnlineDriveObject	695
7.28.3.16	OnlineDriveObjectContainer	695
7.28.3.17	StartDriveDownloadCheckConfiguration	696
7.28.3.18	SafetyTelegram	696
7.28.3.19	Telegram	697
7.28.3.20	TelegramComposition	698
7.28.3.21	TelegramType	699
7.28.3.22	TorqueTelegram	699
7.28.4	Code examples	700
7.28.4.1	Determining the activation status	700
7.28.4.2	Executing drive functions	701
7.28.4.3	Creating a drive unit	701
7.28.4.4	Creating a drive component	702
7.28.4.5	Determining a drive object	703
7.28.4.6	Determining the drive object type	703
7.28.4.7	Reading and writing BICO parameters	704
7.28.4.8	Download	705
7.28.4.9	Editing DRIVE-CLiQ connections	707
7.28.4.10	Carrying out the first steps in Startdrive	708
7.28.4.11	Defining the encoder type	709
7.28.4.12	Configuring devices	716
7.28.4.13	Creating a component for a drive component (S120 only)	717
7.28.4.14	Defining the motor type and motor configuration	718
7.28.4.15	Reading and writing parameters	720
7.28.4.16	Reading and writing parameters online	722
7.28.4.17	Saving the parameterization	722
7.28.4.18	Using Safety Integrated telegrams	723
7.28.4.19	Inserting and extending telegrams	724
7.28.4.20	Using torque telegrams	725
7.28.4.21	Restoring factory settings	726
7.29	Functions for DCC	726
7.29.1	Introduction	726
7.29.2	DCC Openness	727
7.29.3	DCC Openness object model	728
7.29.4	References	728
7.29.4.1	DriveControlChartContainer	728

7.29.4.2	DriveControlChartComposition	729
7.29.4.3	DcclImportOptions	730
7.29.4.4	DcclImportResultData	730
7.29.4.5	DriveControlChart	730
7.29.4.6	Importing DCB extension library	731
7.29.5	Code examples	732
7.29.5.1	General	732
7.29.5.2	Accessing a DriveControlChartContainer via DriveObject	732
7.29.5.3	Retrieving charts	732
7.29.5.4	Accessing charts	732
7.29.5.5	Importing charts	733
7.29.5.6	Exporting charts	733
7.29.5.7	Export all charts	733
7.29.5.8	Retrieving charts in the order in which they are executed	734
7.29.5.9	Finding charts based on their names	734
7.29.5.10	Displaying an import report	734
7.29.5.11	Deleting a chart	735
7.29.5.12	Optimizing the execution sequence	735
7.29.5.13	Importing a DCB Extension library	736
7.29.6	DCC Openness exceptions	737
7.29.6.1	Exception handling	737
7.29.6.2	DriveControlChart Exceptions	738
7.29.6.3	DriveControlChartComposition exceptions	739
7.29.6.4	ImportDcbLibrary Exceptions	740
7.30	Exceptions	740
7.30.1	Handling exceptions	740
7.30.2	Custom exception	743
8	Export/import	747
8.1	Overview	747
8.1.1	Basic principles of importing/exporting	747
8.1.2	Field of application for Import/Export	750
8.1.3	Version Specific Simatic ML Import	750
8.1.4	Editing the XML file	751
8.1.5	Exporting configuration data	752
8.1.6	Importing configuration data	753
8.2	Import/export of project data	755
8.2.1	Project graphics	755
8.2.1.1	Exporting/importing graphics	755
8.2.1.2	Exporting all graphics of a project	756
8.2.1.3	Importing graphics to a project	757
8.2.2	Project texts	758
8.2.2.1	Export of project texts	758
8.2.2.2	Import of project texts	759
8.3	Importing/exporting data of an HMI device	761
8.3.1	Structure of an XML file	761
8.3.2	Structure of the data for importing/exporting	763
8.3.3	Cycles	766
8.3.3.1	Exporting cycles	766
8.3.3.2	Importing cycles	767
8.3.4	Tag tables	768

8.3.4.1	Exporting HMI tag tables	768
8.3.4.2	Importing HMI tag table	771
8.3.4.3	Exporting an individual tag from an HMI tag table	772
8.3.4.4	Importing an individual tag into an HMI tag table	773
8.3.4.5	Special considerations for the export/import of HMI tags	773
8.3.5	VB scripts	775
8.3.5.1	Exporting VB scripts	775
8.3.5.2	Exporting VB scripts from a folder	776
8.3.5.3	Importing VB scripts	777
8.3.6	Text lists	778
8.3.6.1	Exporting text lists from an HMI device	778
8.3.6.2	Importing a text list into an HMI device	779
8.3.6.3	Advanced XML formats for export/import of text lists	780
8.3.7	Graphic lists	782
8.3.7.1	Exporting graphic lists	782
8.3.7.2	Importing a graphic list	782
8.3.8	Connections	783
8.3.8.1	Exporting connections	783
8.3.8.2	Importing connections	784
8.3.9	Screens	785
8.3.9.1	Overview of exportable screen objects	785
8.3.9.2	Exporting all screens of an HMI device	789
8.3.9.3	Exporting a screen from a screen folder	790
8.3.9.4	Importing screens to an HMI device	792
8.3.9.5	Exporting permanent areas	794
8.3.9.6	Importing permanent areas	795
8.3.9.7	Exporting all screen templates of an HMI device	796
8.3.9.8	Exporting screen templates from a folder	797
8.3.9.9	Importing screen templates	799
8.3.9.10	Exporting a pop-up screen	800
8.3.9.11	Importing a pop-up screen	801
8.3.9.12	Exporting a slide-in screen	803
8.3.9.13	Importing a slide-in screen	804
8.3.9.14	Exporting a screen with a faceplate instance	805
8.3.9.15	Importing a screen with a faceplate instance	807
8.4	Export/Import Watch & Force Table	810
8.5	Importing/exporting data of a PLC device	812
8.5.1	Blocks	812
8.5.1.1	XML structure of the block interface section	812
8.5.1.2	Changes of the object model and XML file format	822
8.5.1.3	Exporting DBs with snapshots	824
8.5.1.4	Exporting blocks with know-how protection	826
8.5.1.5	Export/Import of SCL blocks	826
8.5.1.6	Export/Import of structured types of SCL blocks	840
8.5.1.7	Export/Import of SCL call blocks	846
8.5.1.8	Export/Import multilingual comments in SCL	863
8.5.1.9	Exporting failsafe blocks	864
8.5.1.10	Exporting system blocks	864
8.5.1.11	Exporting GRAPH blocks with multi-language text	865
8.5.1.12	Importing block	866
8.5.1.13	Exporting blocks	868

8.5.1.14	Importing blocks/UDT with open reference	875
8.5.1.15	Importing blocks/UDT for structural change object	876
8.5.1.16	Export/Import of unit specific publishing attribute of blocks and types	878
8.5.2	Technology objects	882
8.5.2.1	S7-1200 Motion Control	882
8.5.2.2	S7-1500 Motion Control	882
8.5.2.3	PID control	882
8.5.2.4	Counting	882
8.5.2.5	Easy Motion Control	882
8.5.3	Tag tables	882
8.5.3.1	Exporting PLC tag tables	882
8.5.3.2	Importing PLC tag table	883
8.5.3.3	Exporting an individual tag or constant from a PLC tag table	884
8.5.3.4	Importing an individual tag or constant into a PLC tag table	885
8.5.4	Exporting user data type	886
8.5.5	Importing user data type	887
8.5.6	Export of data in OPC UA XML format	888
8.6	Importing/exporting hardware data	889
8.6.1	AML file format	889
8.6.2	Pruned AML	890
8.6.3	Overview of the objects and parameters of the CAx import/export	891
8.6.4	Structure of the CAx data for importing/exporting	893
8.6.5	AML type identifiers	898
8.6.6	Export/Import of base unit Information via AML	901
8.6.7	Export/Import AML with extension rack connection	904
8.6.8	Connection handling for extension racks	910
8.6.9	Export of CAx data	911
8.6.10	Import of CAx data	914
8.6.11	Export/Import of sub modules	916
8.6.12	Import CAx data without logical address	924
8.6.13	Exceptions during import and export of CAx data	931
8.6.14	Round trip exchange of devices and modules	932
8.6.15	Import AML ignoring unknown artifacts	935
8.6.16	Export/Import topology	940
8.6.17	Import of device with Library references	941
8.6.18	Export of a device object	943
8.6.19	Import of a device object	947
8.6.20	Export/Import of device with set address	950
8.6.21	Export/Import of device with channels	953
8.6.22	Export of device item objects	955
8.6.23	Import of device item objects	960
8.6.24	Export/Import of GSD/GSDML based devices and device items	963
8.6.25	Export/Import device configuration with virtual interface	969
8.6.26	Export/Import of subnets	972
8.6.27	Export/Import of IO-systems	979
8.6.28	Export/Import of multilingual comments	980
8.6.29	AML AR APC 1.1 Attribute Support	982
8.6.29.1	Export/Import of PLC tags	982
8.6.29.2	Export/Import of RH/PLC	985
8.6.29.3	Export/Import AML with customized tag and deviceitems	987
8.6.29.4	Export/Import of FailSafe PLC	990
8.6.29.5	Export/Import of Failsafe IO	996

8.6.29.6	Export/Import vendor specific attribute	997
8.6.30	AML attributes versus TIA Portal Openness attributes	998
9	Major Changes	1003
9.1	Major changes in TIA Portal Openness V15.1	1003
9.2	Major changes in TIA Portal Openness V15	1005
9.3	Major changes in V14 SP1.....	1008
9.3.1	Major changes in V14 SP1.....	1008
9.3.2	Major changes in the object model	1011
9.3.3	Changes on pilot functionality	1015
9.3.4	Changes for export and import.....	1020
9.3.4.1	Changes for export and import.....	1020
9.3.4.2	Changes in API	1020
9.3.4.3	Schema extension.....	1021
9.3.4.4	Schema changes	1024
9.3.4.5	Behaviour changes	1027
9.3.4.6	Block attribute changes.....	1038
9.4	Major changes in V14	1040
9.4.1	Major changes of the object model	1040
9.4.2	Before updating an application to TIA Portal Openness V14	1042
9.4.3	Major string changes.....	1042
9.4.4	Import of files generated with TIA Portal Openness V13 SP1 and previous	1046
Index.....		1049

Security note

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines, equipment and/or networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions only form one element of such a concept.

Customer is responsible to prevent unauthorized access to its plants, systems, machines and networks. Systems, machines and components should only be connected to the enterprise network or the internet if and to the extent necessary and with appropriate security measures (e.g. use of firewalls and network segmentation) in place.

Additionally, Siemens' guidance on appropriate security measures should be taken into account. For more information about industrial security, please visit

<http://www.siemens.com/industrialsecurity>

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends to apply product updates as soon as available and to always use the latest product versions. Use of product versions that are no longer supported, and failure to apply latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under

<http://www.siemens.com/industrialsecurity>

Readme TIA Portal Openness

2.1 Readme

Security measures for TIA Portal Openness applications

It is recommended

- to install a TIA Portal Openness application with admin rights to the programs folder.
- to avoid the dynamical loading of program parts like assemblies or dlls from the users area.
- to run the TIA Portal Openness application with user rights.

Hardware parameters

A description of hardware parameters is available in the installation folder of TIA Portal at
Siemens\Automation\Portal V*\PublicAPI\V*\HW Parameter Description
\Openness_hardware_parameter_description.pdf

Note

V* refers to adapted path according to the installed version of TIA Portal.

Copying a TIA Portal Openness application

When you copy an executable TIA Portal Openness application, it may occur under certain circumstances that the directory path in which the TIA Portal Openness application was originally created is read out by the TIA Portal Openness application.

Remedy:

If you have copied the TIA Portal Openness application to a new directory, open and close the properties dialog to update the Windows cache.

Support of specific features in a TIA Portal project

Multiuser

TIA Portal Openness doesn't support administrative multiuser operations. Because of that it's not recommended to use TIA Portal Openness in Multiuser projects. Be aware that there are TIA Portal Openness actions that actually interfere with the multiuser workflow that is enforced by the GUI of the TIA Portal. If you want to make modifications with TIA Portal Openness, export the multiuser project to a single user project before.

2.1 Readme

Failsafe

When you are using TIA Portal Openness there are restrictions regarding failsafe. Please consider the documentation "SIMATIC Safety - Configuring and Programming" for further information.

Improvement of the TIA Portal Openness performance

To achieve the maximum performance of TIA Portal Openness you can switch off the global search feature of the TIA Portal. To switch off the global search use the GUI or the TIA Portal Openness API call. When the TIA Portal Openness application is finished the global search could be switched on again. Although this is improving the performance, all TIA Portal Openness features work fine even with global search switched on.

Thread-safe program code

Take care that your code is thread-safe, an event appears in a different thread.

Export behaviour of screen items with style enabled

Export of a screen items with style enabled will not export the attributes of the style item, but those of the screen item before activating the style. If a style is selected and UseDesignColorSchema for the screen item is checked, the screen item fetches the attribute values from the style in the user interface but the attribute values of the screen item that were set before selecting the style are still stored in the database for this screen item. TIA Portal Openness exports these actual values that are stored in the database.

After disabling and enabling the style and exporting the screen item again, the same attribute values will be exported for the screen item like in the style item. If UseDesignColorSchema is unchecked, the attribute values of the selected style item are saved to the database for that screen item.

This problem can be solved by following the steps below:

1. Associate the screen item to the style item:
 - The database contains the attribute values before activating the style.
 - The user interfaces fetches attributes from the style item directly.
2. Export the screen item associated to the style item:
 - The XML file contains the attribute values from the database which are those before activating style.
3. Disable the UseDesignColorSchema:
 - The attribute values of style item are written in the attributes of the screen item in the database.

4. Enable the UseDesignColorSchema:
 - The attribute values of the screen item in the database are not changed and are still the ones from 3.
 - The user interfaces fetches attributes from the style item directly.
5. Export the screen item associated to the style item:
 - The XML file contains the attribute values from the database which were set at step 3, which are the same as the values in the style item.

Copying S7-1500 Motion Control technology objects

A copy of TO_CamTrack, TO_OutputCam or TO_MeasuringInput from the project library or global library to the project is not possible.

Importing ASi slaves via AML

If one of the following ASi slaves is imported via an aml-file the firmware version of the device item will be set to V13.0 in all cases:

- ASIsafe FS400 RCV-B: 3SF7 844-*B***-***1
- ASIsafe FS400 RCV-M: 3SF7 844-*M***-***1
- ASIsafe FS400 TRX-M: 3SF7 844-*M***-**T0
- ASIsafe FS400 RCV-C: 3SF7 844-*T***-***1

Exporting and importing function keys

Function keys are synchronized during the import. If a function key is created in the global screen and the key is empty in the screen, the corresponding function key will use the global definition in all screens.

If you want to disable the global use of function keys after the import, define empty keys in the screens and import the screen types in the following order: Global screen, templates, screens.

If you want to ensure when exporting the screens that the global definition of a function key is not used by the template or by the global screen, create an empty function key in the screen. Select the required function key in the screen, then enable the "Use global assignment" property and disable it again.

Accessing a device while Online

Writing attributes of a device that is Online is not supported. Reading attributes is supported.

Disconnecting a subnet is not supported when the device is online.

Instance-specific attributes when importing blocks via TIA Portal Openness

In certain situations, the import rules can mean the loss of instance-specific attributes, such as start values, for example.

Information on specific features

Please See FAQ entries in Siemens Industry Online Support for further information concerning the following Openness functionalities:

- Archive/retrieve project
- Export/import watch tables

2.2 Major changes in TIA Portal Openness V16

Changes

If you have considered the hints concerning programming across versions and you do not upgrade your project to V16 your applications will run without any restrictions on any computer even if only a TIA Portal V16 is installed.

If you upgrade your project to V16 it is necessary to recompile your application using the SiemensEngineering.dll of V16. In some cases it might be necessary to adapt the code of your application

Export of datablocks with export option "None"

As of TIA Portal Openness V16 members of a datablock which are read only will be exported as informative items. It is not longer possible to change these attributes in the exported xml.

I&M and Profisafe address attributes of pushbutton panel and key panel

As of TIA Portal Openness V16 I&M and Profisafe address attributes of pushbutton panel and key panel will be accessible at module level.

Export DB attributes of memory layout property

As of TIA Portal Openness V16 IDB of FBs, ArrayDBs and Graph blocks attributes of memory layout property will be exported with `ReadOnly="True"`.

Unsupported attribute in units

As of TIA Portal Openness V16 inconsistent import will support unit blocks and UDTs with attribute "Access" but not tag tables.

SimaticML XML file's schema definition

As of TIA Portal Openness V16 boolean attribute node's SystemDefined attribute's default value become false in SimaticML XML file's schema definition. However, the change doesn't affect any XML export/ import feature. It is only important for users who try to generate XMLs using the schema file.

Enhance XML import method of blocks and types

As of TIA Portal Openness V16 extend the new import method with additional parameters (i.e. path, importOptions, and swlImportOptions) and a new enum value IgnoreUnitAttribute. By using swlImportOptions.IgnoreUnitAttributes, you can import block from unit to non-unit composition.

AML GUIDs are stored in App IDs

As of TIA Portal Openness V16 AML GUIDs are stored in CustomIdentity (App IDs) attribute instead of Comment to support round trip exchange of device and module.

Name of HMI Object classes

As of TIA Portal Openness V16, the following table shows the list of required name changes for HMI Object classes:

Class Name	New Class Name
AnalogAlarmComposition	HmiAnalogAlarmComposition
DiscreteAlarmComposition	HmiDiscreteAlarmComposition
AlarmClassComposition	HmiAlarmClassComposition
DataLogComposition	HmiDataLogComposition
AlarmLogComposition	HmiAlarmLogComposition
LoggingTagComposition	HmiLoggingTagComposition
LogConfiguration	DataLog

Addition/Removal of properties name for DataLog/Alarm Log

As of TIA Portal Openness V16, the following new properties "StorageDevice" and "StorageFolder" are added to DataLog/AlarmLog and properties such as "StoragePath" and "RequireExplicitRelease" are removed from DataLog/AlarmLog and ScreenItems respectively.

You can find the updated code example for StorageDevice and StorageFolder properties as below:

```
HmiDataLog dataLog = hmiSoftware.DataLogs.Find("DataLog1");
dataLog.Settings.StorageDevice = DeviceNode.Local;
dataLog.Settings.StorageFolder = @"D:\workdir\DataLogs";
```

Remove a property name from HMI Tag

As of TIA Portal Openness V16, the 'DisplayName' property is removed from the HMI Tag object.

2.3 Announcement of major changes in future releases

Announcement of changes

The TIA Portal Openness API will be changed in a later version. There is no need to change the code of your application immediately, because applications based on former releases will run without any restriction. But for new applications it is recommended to use the new functionality and to plan the recoding of your application, as of V17 the following methods will not be supported any longer.

- Type of compositions and associations

Type of compositions and associations

The following types will be changed to indicate the snapshot behaviour:

- AddressAssociation
- AddressComposition
- AddressControllerAssociation
- ChannelComposition
- DeviceItemAssociation
- DeviceItemComposition
- HwIdentifierAssociation
- HwIdentifierComposition
- HwIdentifierControllerAssociation
- IoConnectorComposition
- IoControllerComposition

Simocode Openness

Please note that while all Truth Table parameters will be available for use in Openness in V15.1, their implementation will change in V16.

Instead of having to set each bit individually, it will be possible to set all outputs bits of a Truth Table in one quick array operation.

If you want to keep using your existing Openness application in V16, you may have to adapt your code to the new situation.

Line conductor of energy meter

Line conductors of an energy meter had been realized as channel. As of V16 will be set up as structures based on complex data types. For new applications it is recommended to access line conductors as structures.

2.4 Hints for writing long-term stable code

Version change

If you consider some hints for writing long-term stable code you will be able to use your application with other versions of the TIA Portal without modifying the code of your application.

Note

V* and *.ap* in document refer to the adapted path and extension according to the installed version of the TIA Portal.

Registry path and appconfig file

Modifications are necessary to change registry path and application configuration file, for instance:

“C:\Program Files\Siemens\Automation\Portal V14\PublicAPI\V14
SP1\Siemens.Engineering.dll”

has to be changed to

“C:\Program Files\Siemens\Automation\Portal V*\PublicAPI\V*\Siemens.Engineering.dll”

To write long-term stable code, the registry path should be configurable and the application configuration file must be updated.

Installation path

Modifications are necessary to change the installation path of TIA Portal, for instance:

“C:\Program Files\Siemens\Automation\Portal V14\PublicAPI\V14
SP1\Siemens.Engineering.dll”

has to be changed to

“C:\Program Files\Siemens\Automation\Portal V*\PublicAPI\V*\Siemens.Engineering.dll”

To write long-term stable code, the installation path should be configurable.

Path of AmiHost

Modifications are necessary to change the path of AmiHost, for instance:

“C:\Program Files\Siemens\Automation\Portal V14\bin
\Siemens.Automation.Cax.AmiHost.exe”

has to be changed to “C:\Program Files\Siemens\Automation\Portal V*\bin
\Siemens.Automation.Cax.AmiHost.exe”

To write long-term stable code, the path of AmiHost should be configurable.

Extensions of TIA Portal project files and libraries

Modifications are necessary to change the extensions of TIA Portal project file and of libraries, for instance:

*.ap14
has to be changed to
.ap

To write long-term stable code, the extensions of TIA Portal project files and libraries should be configurable.

Opening a project

To write long-term stable code, the `Projects.OpenWithUpgrade` method should be used instead of the `Projects.Open` method.

Hierarchy of compare, compile or download results

The hierarchy and/or the order of compare, compile or download results might change across versions.

To write long-term stable code, you should avoid making assumptions about the depth and order of specific results.

The class layout is actually considered long term stable, mention explicit type names `CompilerResult`, `CompareResult`, `DownloadResult`, `UploadResult`. There is also a new results class: `UploadResult`. Content, hierarchy and order follow what is presented on the TIA Portal user interface of the currently executed or installed TIA Portal.

What's new in TIA Portal Openness?

Compatibility and long term stability

- Siemens.Engineering.dll assemblies
Because the Siemens.Engineering.dll assemblies of V14 SP1, V15, V15.1 and V16 are included in the scope of delivery, applications based on V14 SP1, V15 and V15.1 also run in V16 without modification. To make use of the functions of V16, you must integrate the DLL of V16 and recompile the application.
The Siemens.Engineering.dll assemblies can be found in the installation directory under "PublicAPI[version]\". For example, the V14 SP1 dll can be found as "C:\Program Files \Siemens\Automation\Portal V*\PublicAPI\V14 SP1\Siemens.Engineering.dll".
- Exporting Simatic ML files
The Siemens.Engineering.dll assemblies V14 SP1, V15, V15.1 and V16 will create Simatic ML files of TIA Portal version V16.
- Importing Simatic ML files
Each Siemens.Engineering.dll assemblies is able to import Simatic ML files from that version and any supported version from a previous release. For example, Simatic ML files of V15 can be imported with Siemens.Engineering.dll assemblies V16. Simatic ML files of V16 can not be imported with Siemens.Engineering.dll assemblies V15.

Note

V* refer to the installed version of the TIA Portal Openness API.

For changes to the object model see Major Changes in TIA Portal Openness V16 for further information.

New features

The following new features and innovations are available in TIA Portal Openness V16. You can find additional details on the various topics in the individual sections of the product documentation.

- User specific flags for identification of devices and modules
- Querying online checksums of PLCs for Hardware and software.
- Support of Software Units.
- Support of Export/Import of technological objects
- Support of Version Control Interface (VCI)
- Extensions for CAx exports and imports concerning
 - Support of APC AR V1.1
 - Using objects from libraries
 - Support of base units of ET200SP

- Extension to the support of hardware configuration concerning
OPC UA Server configuration and user management
Management of certificates
Web Server configuration and user management
Watch tables for Web Server and display
- Support of the configuration of WinCC Unified
- Support of SiVArc concerning
Creation and modification of SiVArc rules
- Support of Safety engineering even if a password is set for the Safety programm

4.1 Requirements for TIA Portal Openness

Requirements for using TIA Openness applications

- A product based on the TIA Portal is installed on the PC, for example, "STEP 7 Professional" or "WinCC Professional".
- The "TIA Portal Openness" is installed on the PC.
See Installing TIA Portal Openness (Page 30)

Supported Windows operating systems

The following table shows which combinations of Windows operating system, TIA Portal and user application are mutually compatible:

Windows operating system	TIA Portal	User application
64-bit	64-bit	32-bit, 64-bit and "Any CPU"

Requirements for programming TIA Portal Openness applications

- Microsoft Visual Studio 2015 Update 1 or later with .Net Framework SDK 4.6.2 and the Windows Classic Desktop package

Necessary user knowledge

- Knowledge as a system engineer
- Advanced knowledge of Microsoft Visual Studio 2015 Update 1 or later with .Net Framework SDK 4.6.2
- Advanced knowledge of C# / VB.net and .Net Framework
- User knowledge of the TIA Portal

TIA Portal Openness remoting channels

The TIA Portal Openness remoting channels are registered as type IpcChannel with the "ensureSecurity" parameter set to "false".

Note

You should avoid registering another IpcChannel using a "ensureSecurity" parameter value other than "false" with a priority higher than or equal to "1".

The IpcChannel is defined with the following attributes:

Attribute	Settings
"name" and "portName"	Set to "\${Process.Name}_{Process.Id}" or "\${Process.Name}_{Process.Id}_{AppDomain.Id}" when registered in an AppDomain other than the application's default.
"priority"	Set with the default value of "1".
"typeFilterLevel"	Set to "Full".
"authorizedGroup"	Set to the NTAccount value string for the built-in user account (i.e. everyone).

See also

[Adding users to the "Siemens TIA Openness" user group \(Page 31\)](#)

4.2 Installation

4.2.1 Installing TIA Portal Openness

Introduction

The "TIA Portal Openness" is installed via TIA Portal setup program by selecting the TIA Portal Openness checkbox (under Options) during TIA Portal installation.

Requirements

- Hardware and software of the programming device or PC meet the system requirements.
- You have administrator rights.
- Running programs are closed.
- Autorun is disabled.
- WinCC and/or STEP 7 are installed.
- The version number of the "TIA Portal Openness" matches the version numbers of WinCC and STEP 7.

Note

If a previous version of TIA Portal Openness is already installed, the current version will be installed side by side.

Procedure

To install the TIA Portal Openness, ensure the TIA Portal Openness checkbox is selected during the installation of TIA Portal. Follow the below steps to check the TIA Openness installation.

1. Under Configuration menu, select the folder Options.
2. Check the TIA Portal Openness checkbox.
3. Click "Next" and select the required option.

Follow the installation procedure of TIA Portal to complete the TIA Portal Openness installation.

Result

"TIA Portal Openness" is installed on the PC. Moreover, the local user group "Siemens TIA Openness" is generated.

Note

You still do not have access to the TIA Portal with the "TIA Portal Openness" add-on package. You need to be a member of the "Siemens TIA Openness" user group (see Adding users to the "Siemens TIA Openness" user group (Page 31)).

4.2.2 Adding users to the "Siemens TIA Openness" user group

Introduction

When you install TIA Portal Openness on the PC, the "Siemens TIA Openness" user group is automatically created.

Whenever you access the TIA Portal with your TIA Portal Openness application, the TIA Portal verifies that you are a member of the "Siemens TIA Openness" user group, either directly or indirectly by way of another user group. If you are a member of the "Siemens TIA Openness" user group, the TIA Portal Openness application starts and establishes a connection to the TIA Portal.

Procedure

You add a user to the "Siemens TIA Openness" user group with applications from your operating system. The TIA Portal does not support this operation.

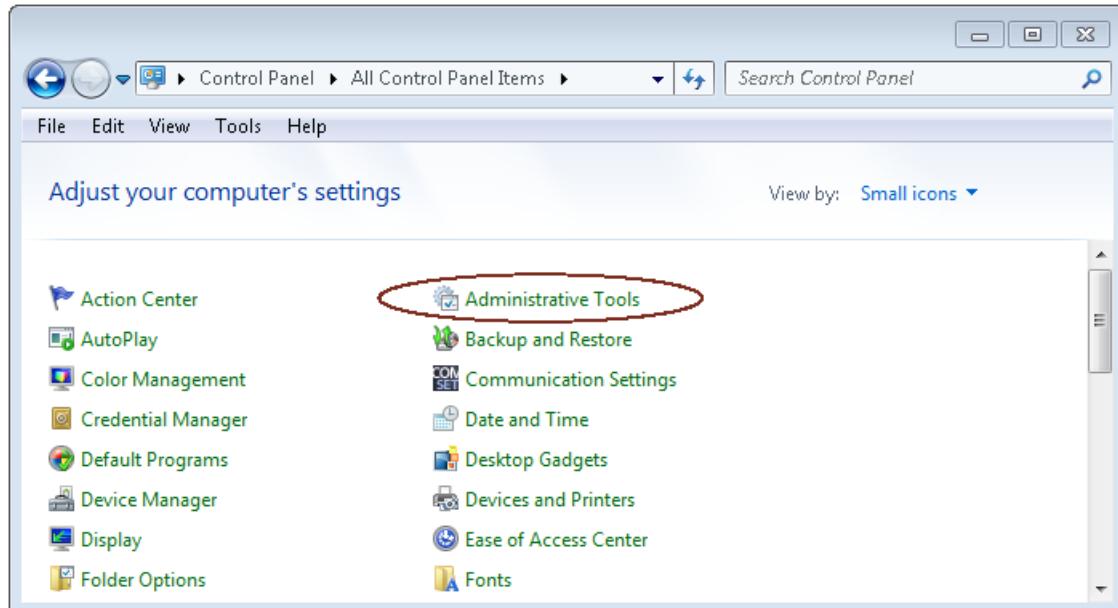
Note

Depending on the configuration of your domain or computer, you may need to log on with administrator rights to expand the user group.

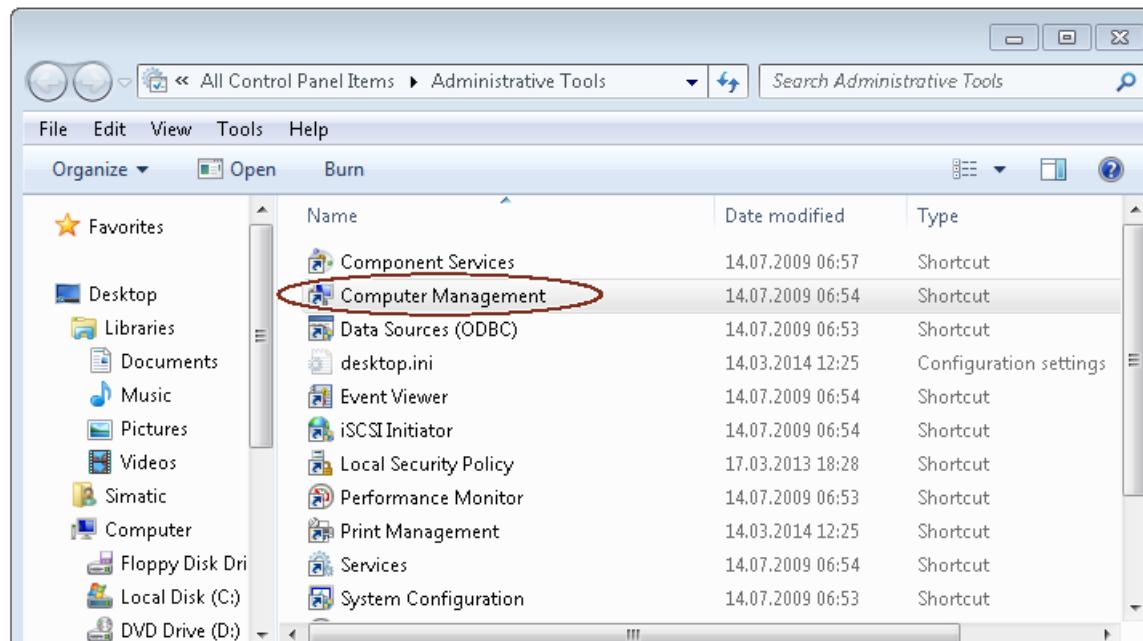
4.2 Installation

In a Windows 7 operating system (English language setting), for example, you can add a user to the user group as follows:

1. Select "Start" > "Control Panel".
2. Double-click "Administrative Tools" in the Control Panel.



3. Click "Computer Management" to open the configuration dialog of the same name.



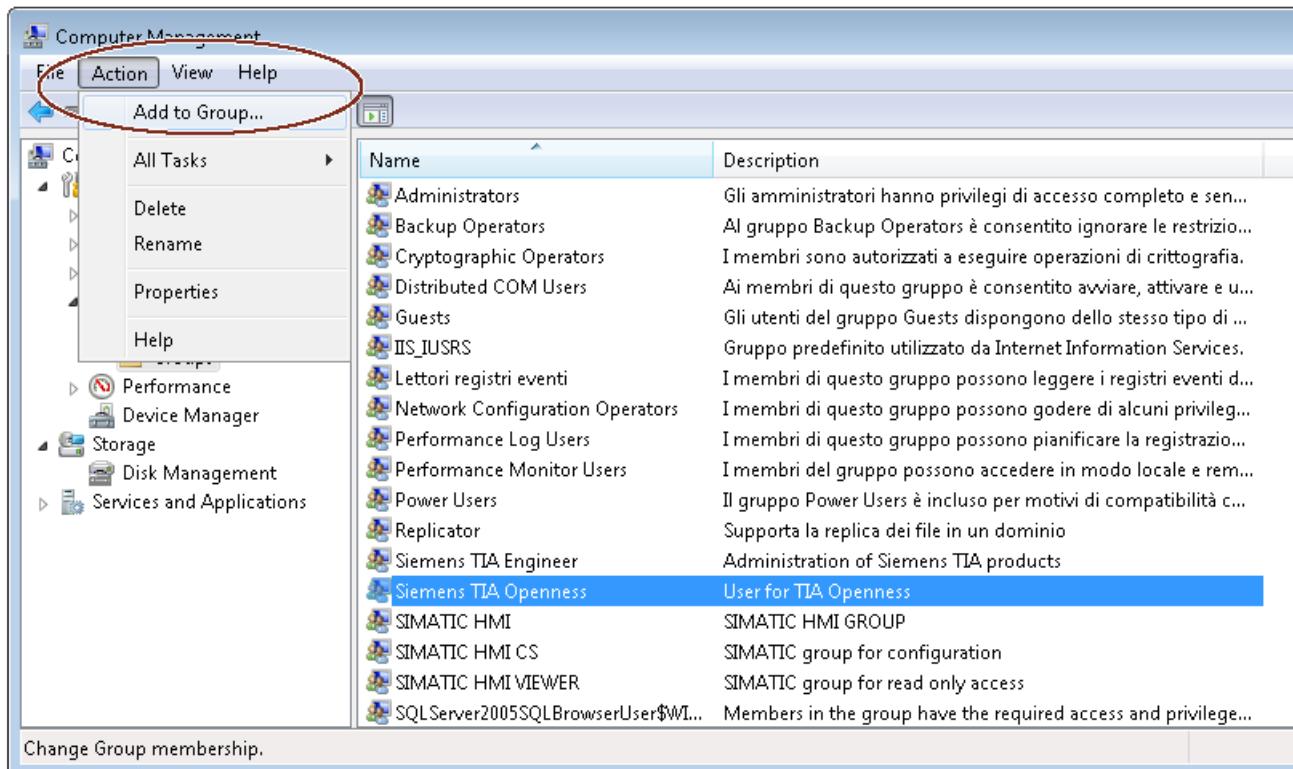
4. Select "Local Users and Groups > Groups", in order to display all created user groups.

5. Select the "Siemens TIA Openness" entry from the list of user groups in the right pane.

The screenshot shows the Windows Computer Management interface. The left navigation pane is collapsed, showing options like System Tools, Local Users and Groups, and Services and Applications. The main pane displays a table of user groups under the heading 'Local Users and Groups'. The columns are 'Name' and 'Description'. A red oval highlights the row for 'Siemens TIA Openness', which is described as 'User for TIA Openness' and 'SIMATIC HMI GROUP'. Other groups listed include Administrators, Backup Operators, Cryptographic Operators, Distributed COM Users, Guests, IIS_IUSRS, Lettori registri eventi, Network Configuration Operators, Performance Log Users, Performance Monitor Users, Power Users, Replicator, Siemens TIA Engineer, and SIMATIC HMI CS, SIMATIC HMI VIEWER.

Name	Description
Administrators	Gli amministratori hanno privilegi di accesso completo e sen...
Backup Operators	Al gruppo Backup Operators è consentito ignorare le restrizion...
Cryptographic Operators	I membri sono autorizzati a eseguire operazioni di crittografia.
Distributed COM Users	Ai membri di questo gruppo è consentito avviare, attivare e us...
Guests	Gli utenti del gruppo Guests dispongono dello stesso tipo di...
IIS_IUSRS	Gruppo predefinito utilizzato da Internet Information Services.
Lettori registri eventi	I membri di questo gruppo possono leggere i registri eventi d...
Network Configuration Operators	I membri di questo gruppo possono godere di alcuni privilegi...
Performance Log Users	I membri di questo gruppo possono pianificare la registratio...
Performance Monitor Users	I membri del gruppo possono accedere in modo locale e remoto...
Power Users	Il gruppo Power Users è incluso per motivi di compatibilità c...
Replicator	Supporta la replica dei file in un dominio
Siemens TIA Engineer	Administration of Siemens TIA products
Siemens TIA Openness	User for TIA Openness SIMATIC HMI GROUP
SIMATIC HMI	SIMATIC group for configuration
SIMATIC HMI CS	SIMATIC group for read only access
SIMATIC HMI VIEWER	

6. Select the "Action > Add to Group..." menu command.

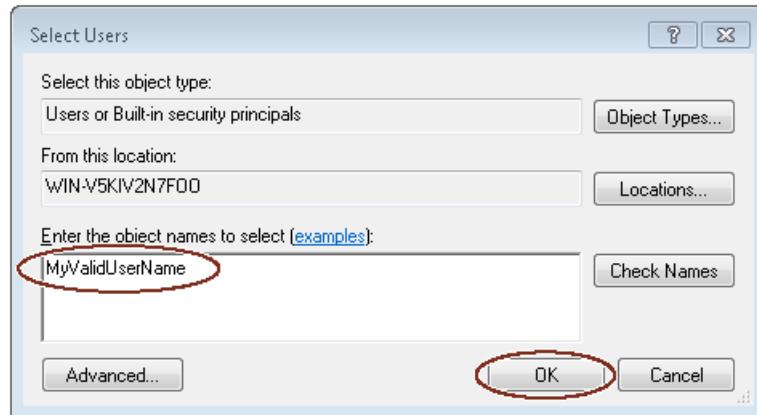


The attributes dialog of the user group opens:



7. Click "Add".

The selection dialog that opens displays the users that can be selected:



8. Enter a valid user name in the input field.

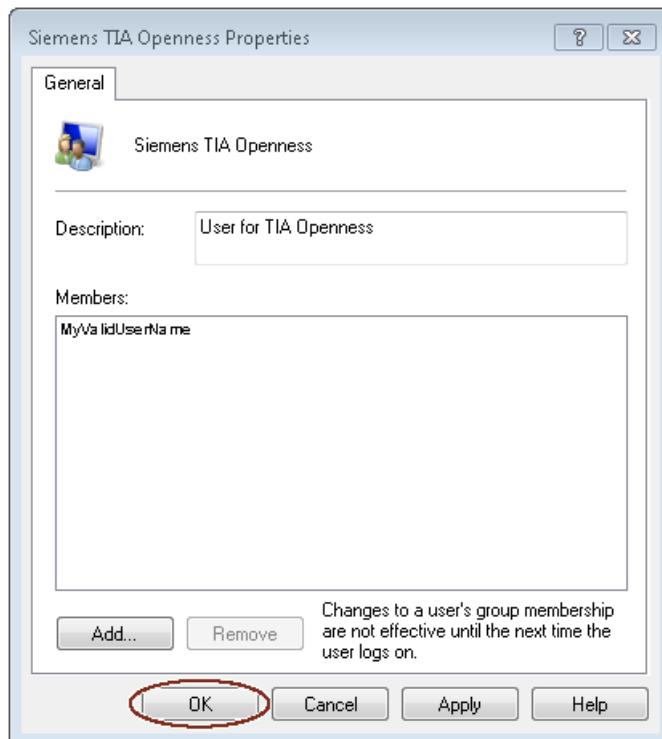
Note

Click "Check Names" to verify that the user entered has a valid user account for this domain or computer.

The "From this location" field displays the domain or computer name for the user name entered. For more information, contact your system administrator.

9. Confirm your selection with "OK".

The new user is now displayed in the attributes dialog of the user group.

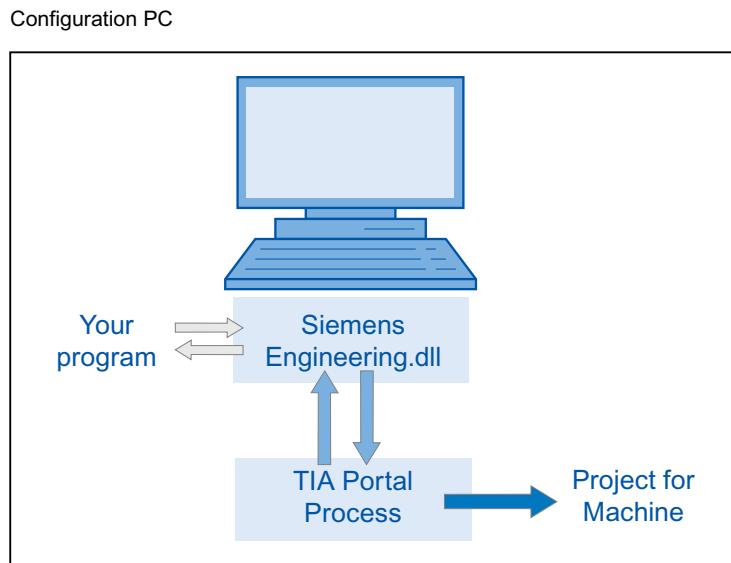


You register additional users by clicking the "Add" button.

10. Click "OK" to end this operation.
11. Log on to the PC again for the changes to take effect.

4.2.3 Accessing the TIA Portal

Overview



Procedure

1. Set up the development environment to access and start the TIA Portal.
2. Instantiate the object of the portal application in your program to start the portal.
3. Find the desired project and open it.
4. Access the project data.
5. Close the project and exit the TIA Portal.

See also

- [Connecting to the TIA Portal \(Page 76\)](#)
[Terminating the connection to the TIA Portal \(Page 85\)](#)

4.3 Openness tasks

4.3.1 Applications

Introduction

TIA Portal Openness provides you with various ways to access the TIA Portal and offers a selection of functions for defined tasks.

You access the following areas of the TIA Portal by using the TIA Portal Openness API interface :

- Portal data
- Project data
- PLC data
- HMI data
- Drive data

Note

You must not use the TIA Portal Openness API to execute checks or generate data for the acceptance/approval of a fail-safe system. Acceptance/approval may only be carried out with a safety printout using the STEP 7 Safety add-on package or with the function test. The TIA Portal Openness API is no substitute.

Accessing the TIA Portal

TIA Portal Openness offers various ways to access the TIA Portal. You create an external TIA Portal instance in the process either with or without UI. You can also access ongoing TIA Portal processes at the same time.

Accessing projects and project data

When accessing projects and project data, you mainly use TIA Portal Openness for the following tasks:

- Close, open and save the project
- Enumerate and query objects
- Create objects
- Delete objects

4.3.2 Export/import

Introduction

TIA Portal Openness supports the import and export of project data by means of XML files. The import/export function supports external configuration of existing engineering data. You use this function to make the engineering process effective and free of error.

Application

You use the import/export function for the following purposes:

- Data exchange
- Copying parts of a project
- External processing of configuration data, for example, for bulk data operations using find and replace
- External processing of configuration data for new projects based on existing configurations
- Importing externally-created configuration data, for example, text lists and tags
- Providing project data for external applications

4.4 Object list

Introduction

The following tables show the available objects up to and including Runtime Advanced, and indicate whether these objects are supported by TIA Portal Openness.

Neither Runtime Professional nor device proxy files are supported by TIA Portal Openness in WinCC.

Objects

You can control the following project data depending on which HMI device you are using:

Table 4-1 Screens

Object	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
Screen	Yes	Yes	Yes	Yes
Global screen	Yes	Yes	Yes	Yes
Templates	Yes	Yes	Yes	Yes
Permanent area	No	Yes	Yes	Yes
Pop-up screen	No	Yes	Yes	Yes
Slide-in screen	No	Yes	Yes	Yes

Table 4-2 Screen objects

Object	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
Line	Yes	Yes	Yes	Yes
Polyline	No	Yes	Yes	Yes
Polygon	No	Yes	Yes	Yes
Ellipse	Yes	Yes	Yes	Yes
Ellipse segment	No	No	No	No
Circle segment	No	No	No	No
Elliptical arc	No	No	No	No
Camera view	No	No	No	No
Circular arc	No	No	No	No
Circle	Yes	Yes	Yes	Yes
PDF view	No	No	No	No
Rectangle	Yes	Yes	Yes	Yes
Connector	No	No	No	No
Text field	Yes	Yes	Yes	Yes
Graphic view	Yes	Yes	Yes	Yes
Pipe	No	No	No	No
Double T-piece	No	No	No	No
T-piece	No	No	No	No
Pipe bends	No	No	No	No
I/O field	Yes	Yes	Yes	Yes
Date/time field	Yes	Yes	Yes	Yes
Graphic I/O field	Yes	Yes	Yes	Yes
Editable text field	No	No	No	No
List box	No	No	No	No
Combo box	No	No	No	No
Button	Yes	Yes	Yes	Yes
Round button	No	No	No	No
Illuminated button	No	No	Yes	No
Switch	Yes	Yes	Yes	Yes
Symbolic I/O field	Yes	Yes	Yes	Yes
Key-operated switch	No	No	Yes	No
Bar	Yes	Yes	Yes	Yes
Symbol library	No	Yes	Yes	Yes
Slider	No	Yes	Yes	Yes
Scroll bar	No	No	No	No
Check box	No	No	No	No
Option buttons	No	No	No	No
Gauge	No	Yes	Yes	Yes
Clock	No	Yes	Yes	Yes
Memory space view	No	No	No	No

4.4 Object list

Object	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
Function keys	Yes	Yes	Yes	Yes
Faceplate instances	No	Yes	Yes	Yes
Screen window	No	No	No	No
User view	Yes	Yes	Yes	Yes
HTML Browser	No	No	No	No
Print job/script diagnostics	No	No	No	No
Recipe view	No	No	No	No
Alarm view	No	No	No	No
Alarm indicator	No	No	No	No
Alarm window	No	No	No	No
Criteria analysis view	No	Yes ¹⁾	Yes	Yes
ProDiag overview	No	Yes ¹⁾	Yes	Yes
GRAPH overview	No	Yes ¹⁾	Yes	Yes
PLC code view	No	Yes ¹⁾	Yes	Yes
f(x) trend view	No	No	No	No
f(t) trend view	No	No	No	No
Table view	No	No	No	No
Value table	No	No	No	No
Media Player	No	No	No	No
Channel diagnostics	No	No	No	No
WLAN reception	No	No	No	No
Zone name	No	No	No	No
Zone signal	No	No	No	No
Effective range name	No	No	No	No
Effective range name (RFID)	No	No	No	No
Effective range signal	No	No	No	No
Charge condition	No	No	No	No
Handwheel	No	No	Yes	No
Help indicator	No	No	No	No
Sm@rtClient view	No	No	No	No
Status/Force	No	No	No	No
System diagnostic view	No	No	No	No
System diagnostic window	No	No	No	No

1) Only Mobile Panels with the device version greater than 12.0.0.0 support this screen object

Table 4-3 Dynamic

Object	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
Display	Yes	Yes	Yes	Yes
Operability	No	Yes	Yes	Yes
Visibility	Yes	Yes	Yes	Yes
Movements	Yes	Yes	Yes	Yes

Table 4-4 Additional objects

Object	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
Groups	Yes	Yes	Yes	Yes
Soft keys	Yes	Yes	Yes	Yes
Cycles	Yes	Yes	Yes	Yes
VB scripts	No	Yes	Yes	Yes
Function lists	Yes	Yes	Yes	Yes
Project graphics	Yes	Yes	Yes	Yes

Table 4-5 Tags

Object	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
Multiplex tags	Yes	Yes	Yes	Yes
Arrays	Yes	Yes	Yes	Yes
User data types	Yes	Yes	Yes	Yes
Internal	No	Yes	Yes	Yes
Points of use of elementary data types	Yes	Yes	Yes	Yes
Points of use of user data types	Yes	Yes	Yes	Yes
Points of use of arrays	Yes	Yes	Yes	Yes

TIA Portal Openness also supports all value ranges which are supported by the communication drivers.

Connections

TIA Portal Openness supports non-integrated connections that are also supported by the respective HMI devices. You can find additional information in the online help for the TIA Portal under "Process visualization > Controller communication > Device-dependent".

Table 4-6 Lists

Object	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
Text lists	Yes	Yes	Yes	Yes
Graphic lists	Yes	Yes	Yes	Yes

Table 4-7 Texts

Object	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
Multilingual texts	Yes	Yes	Yes	Yes
Formatted texts and their instances	No	Yes	Yes	Yes

4.5 Setting attributes of Blocks, DBs & UDTs

Introduction

It is possible to write and read (general) attributes of the Blocks (written in any programming language), DBs and UDTs via the Openness API.

TechnologicalObjects are internally DBs, so this feature is also valid for TOs.

The same checking rules are to be applied on Name by using SetAttribute of the Openness API on block, DB (TO) or UDT which are applied in the GUI.

In case of rule violation an appropriate user exception will be thrown and the attribute value won't be changed.

This extension of the API doesn't affect the XML export / import.

General Attributes

Legend

- X: The attribute is relevant, write to xml, accessible with API
- -: The attribute is not accessible, the attribute does not exist

Attributes	Datatype	Availability via the API		
		CodeBlock	DB	UDT
AutoNumber	Boolean	X (xml); RW (API)	X (xml); RW (API)	-
CodeModifiedDate	DateTime	X (xml); R (API)	X (xml); R (API)	-
CompileDate	DateTime	X (xml); R (API)	X (xml); R (API)	-
CreationDate	DateTime	X (xml); R (API)	X (xml); R (API)	X (xml); R (API)
HeaderAuthor	String	X (xml); R (API)	X (xml); R (API)	-
HeaderFamily	String	X (xml); R (API)	X (xml); R (API)	-
HeaderName	String	X (xml); R (API)	X (xml); R (API)	-
HeaderVersion	System.Version	X (xml); R (API)	X (xml); R (API)	-
InstanceOfName	String	-	X (xml); R (API)	-
Interface	String	X (xml); - (API)	X (xml); - (API)	X (xml); - (API)
InterfaceModifiedDate	DateTime	X (xml); R (API)	X (xml); R (API)	X (xml); R (API)
IsConsistent	Boolean	X (xml); R (API)	X (xml); R (API)	X (xml); R (API)

Attributes	Datatype	Availability via the API		
		CodeBlock	DB	UDT
IsKnowHowProtected	Boolean	X (xml); R (API)	X (xml); R (API)	X (xml); R (API)
MemoryLayout (sofar: Optimization)	MemoryLayout	X (xml); R (API)	X (xml); R (API)	-
ModifiedDate	DateTime	X (xml); R (API)	X (xml); R (API)	X (xml); R (API)
Name	String	X (xml); RW (API)	X (xml); RW (API)	X (xml); RW (API)
Number	Int32	X (xml); RW (API)	X (xml); RW (API)	-
ParameterModified	DateTime	X (xml); R (API)	X (xml); R (API)	-
ProgrammingLanguage	ProgrammingLanguage	X (xml); R (API)	X (xml); R (API)	-
SecondaryType	String	X (xml); R (API)	-	-
StructureModified	DateTime	X (xml); R (API)	X (xml); R (API)	-

4.6 Standard libraries

Insert the following namespace statements at the beginning of the relevant code example to ensure that the code examples work:

```
using System;
using System.Collections.Generic;
using System.IO;
using Siemens.Engineering;
using Siemens.Engineering.Cax;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Compare;
using Siemens.Engineering.Download;
using Siemens.Engineering.Hmi;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.RuntimeScripting;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Extensions;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.HW.Utilities;
using Siemens.Engineering.Library;
using Siemens.Engineering.Library.MasterCopies;
using Siemens.Engineering.Library.Types;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.ExternalSources;
using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.TechnologicalObjects;
using Siemens.Engineering.SW.TechnologicalObjects.Motion;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Upload;
```

4.7 Notes on performance of TIA Portal Openness

Root objects

You can specify several root objects in import files.

Example: You can create several text lists in an XML file instead of one text list for each XML file.

TIA Portal Openness functions

The first call of a TIA Portal Openness function can take longer than any subsequent calls of the TIA Portal Openness function.

Example: If you perform multiple configuration data exports one after the other, the first export can take longer than the subsequent exports.

Introduction

Introduction

TIA Portal Openness describes open interfaces for engineering with the TIA Portal. You will find further information about "TIA Portal Openness - Efficient generation of program code using code generators" in the SIEMENS YouTube channel.

You automate the engineering with TIA Portal Openness by controlling the TIA Portal externally from a program you have created.

You can perform the following actions with TIA Portal Openness:

- Create project data
- Modify projects and project data
- Delete project data
- Read in project data
- Make projects and project data available for other applications.

Note

Siemens is not liable for and does not guarantee the compatibility of the data and information transported via these interfaces with third-party software.

We expressly point out that improper use of the interfaces can result in data loss or production downtimes.

Note

The code snippets contained in this documentation are written in C# syntax.

Due to the shortness of the used code snippets the error handling description has been shortened as well.

Application

The TIA Portal Openness interface is used to do the following:

- Provide project data.
- Access the TIA Portal process.
- Use project data.

Using defaults from the field of automation engineering

- By importing externally generated data
- By remote control of the TIA Portal for generating projects

Providing project data of the TIA Portal for external applications

- By exporting project data

Ensuring competitive advantages through efficient engineering

- You do not have to configure existing engineering data in the TIA Portal.
- Automated engineering processes replace manual engineering.
- Low engineering costs strengthen the bidding position as compared to the competition.

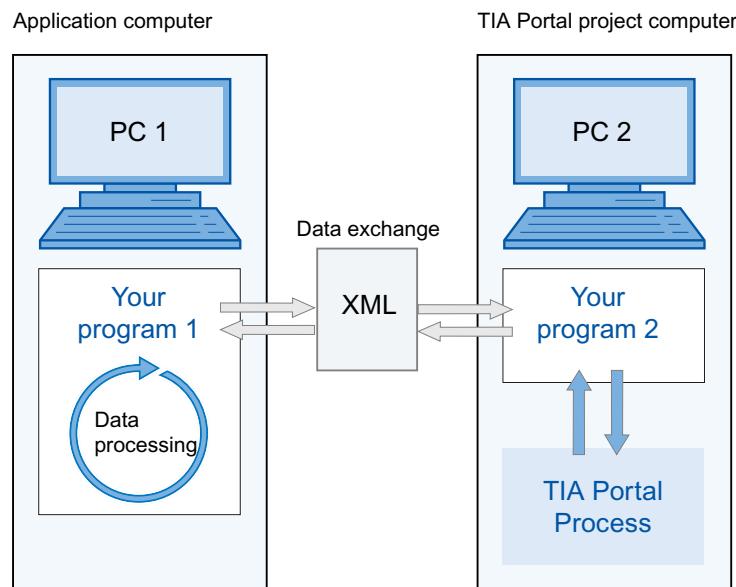
Working together on project data

- Test routines and bulk data processing can take place in parallel to the ongoing configuration.

Configurations

You can work with two variants of the TIA Portal Openness:

Application and the TIA Portal are on different computers



- Data exchange takes place by XML files. The XML files can be exported or imported by your programs.
- The data exported from the TIA Portal project to PC2 can be modified on PC1 and re-imported.

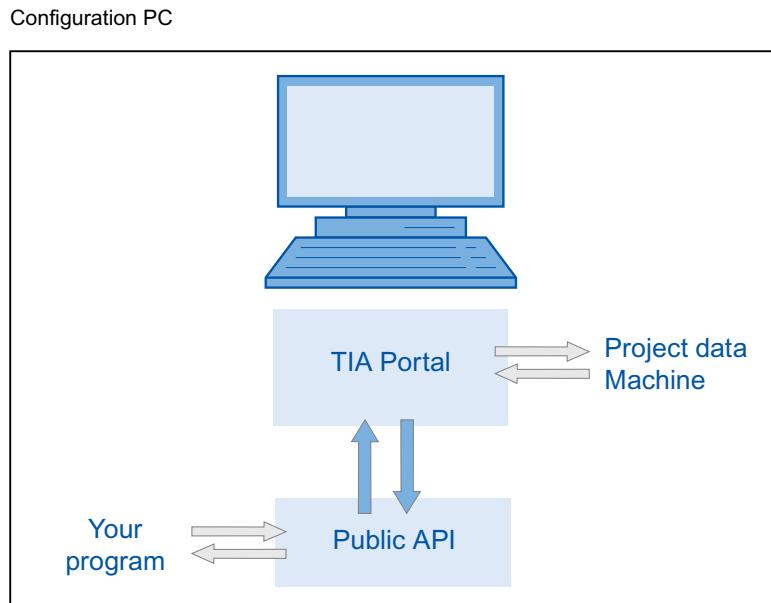
Note

You have to develop an executable program "Your Program 2" for PC2, such as "program2.exe". The TIA Portal runs with this program in the background.

Import and export of XML files takes place exclusively via the TIA Portal Openness API.

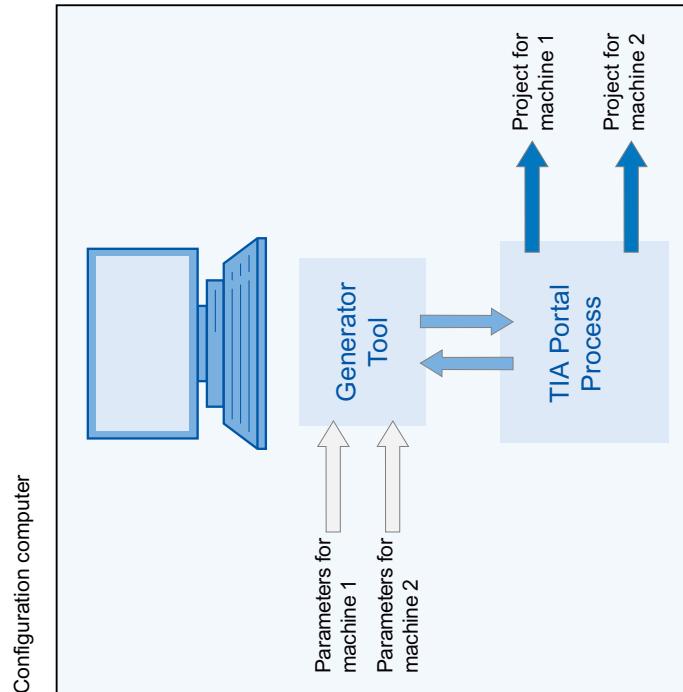
- You can archive exchanged files for verification purposes.
- Exchanged data can be processed at different locations and times.

Application and the TIA Portal are on the same computer



- Your program launches the TIA Portal either with or without the user interface. Your program opens, saves and/or closes a project. The program can also connect to a running TIA Portal.
- You can then use the TIA Portal functionality to request, generate and modify project data or to initiate import or export processes.
- The data is created under control of the TIA Portal processing and stored in the project data.

Typical application in modular mechanical engineering



- An efficient automation system is to be applied to similar machines.
- A project is available in the TIA Portal, which contains the components of all machine variants.
- The Generator tool controls the creation of the project for a specific machine variant.
- The Generator tool obtains the defaults by reading in the parameters for the requested machine variant.
- The Generator tool filters out the relevant elements from the overall TIA Portal project, modifies them if necessary and generates the requested machine project.

TIA Portal Openness API

7.1 Introduction

Overview

TIA Portal Openness supports a selection of functions for defined tasks that you can call outside the TIA Portal by means of the TIA Portal Openness API.

Note

If a previous version of TIA Portal Openness is already installed, the current version will be installed side by side.

You are provided with an overview of the typical programming steps in the sections below. You can learn how the individual code sections interact and how to integrate the respective functions into a complete program. You also get an overview of the code components that have to be adapted for each task.

Example program

The individual programming steps are explained using the "Creating API access in a console application" function as an example. You integrate the provided functions in this program code and adapt the respective code components for this task.

Functions

The section below lists the functions for defined tasks that you can call with TIA Portal Openness outside the TIA Portal.

7.2 Programming steps

Overview

TIA Portal Openness requires the following programming steps for access by means of the TIA Portal Openness API:

1. Make the TIA Portal known in the development environment
2. Set up program access to the TIA Portal
3. Activate program access to the TIA Portal
4. Publish and start the TIA Portal
5. Open project

7.3 TIA Portal Openness object model

6. Execute commands
7. Save and close the project
8. Terminate the connection to the TIA Portal

Note

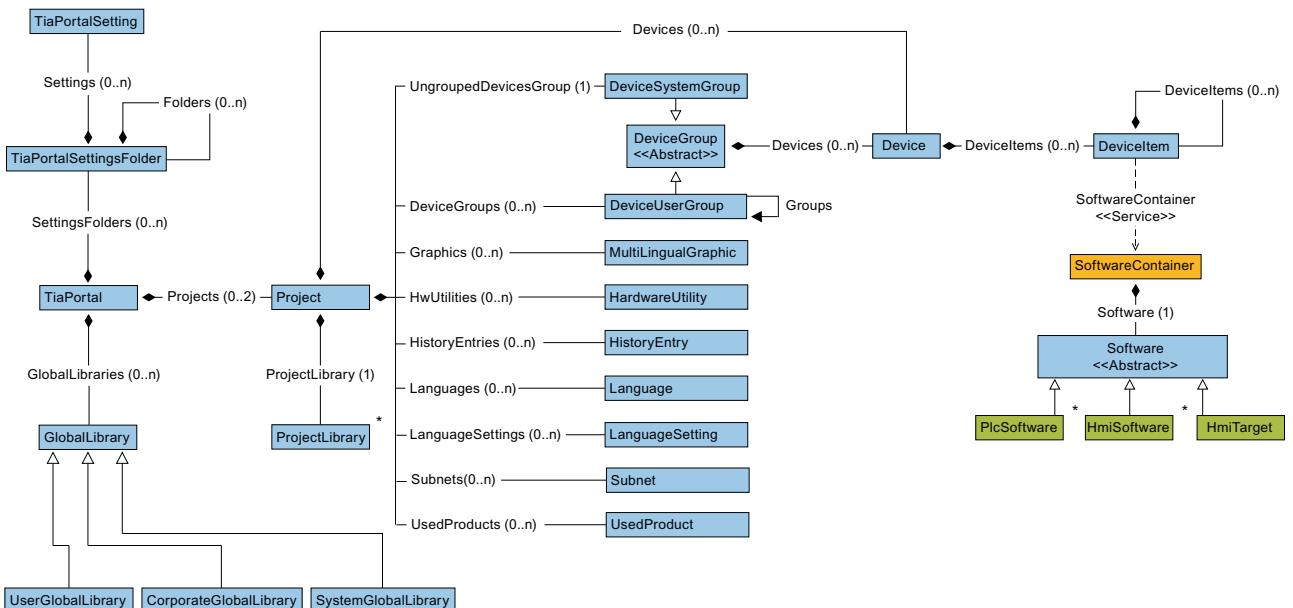
Permitted strings

Only certain characters are allowed in strings in the TIA Portal. All strings passed to the TIA Portal via the TIA Portal Openness application are subject to these rules. If you pass an invalid character in a parameter, an exception is thrown.

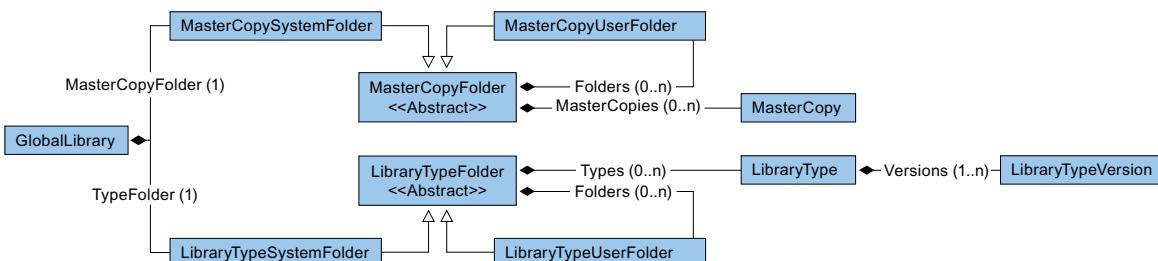
7.3 TIA Portal Openness object model

Overview

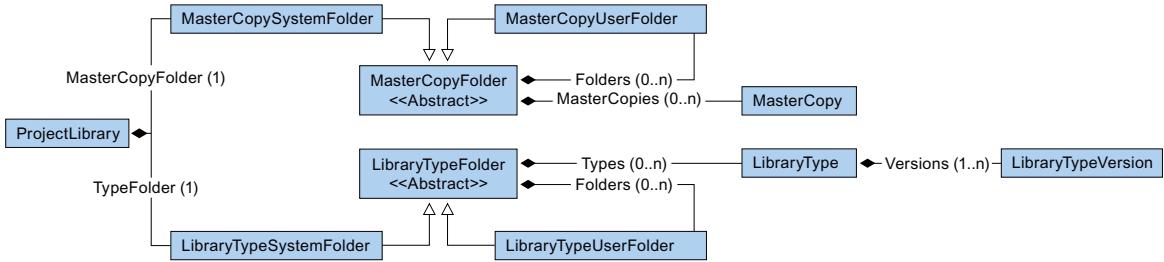
The following diagram describes the highest level of the TIA Portal Openness object model:



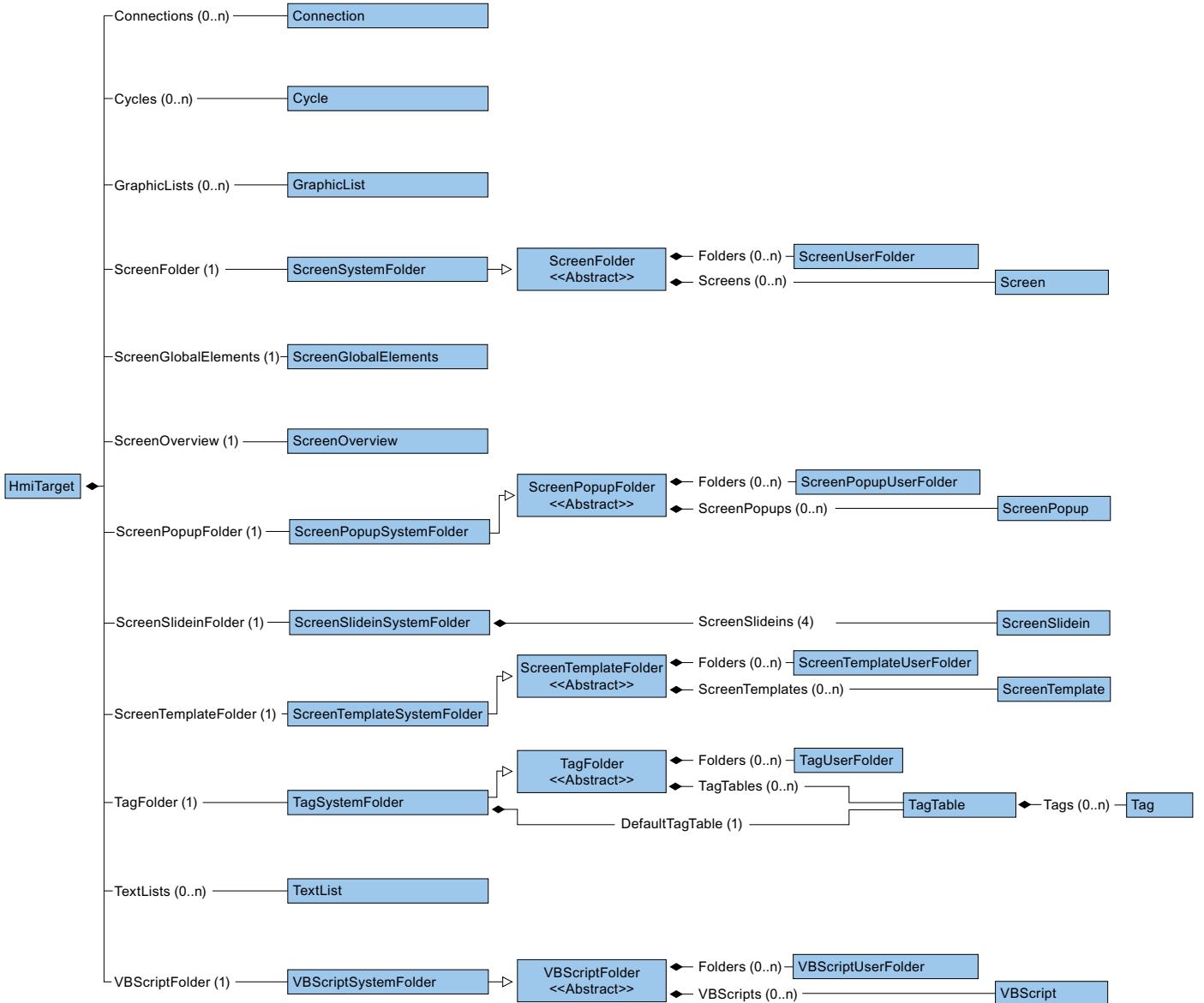
The following diagram describes the objects which are located under GlobalLibrary.



The following diagram describes the objects which are located under ProjectLibrary.

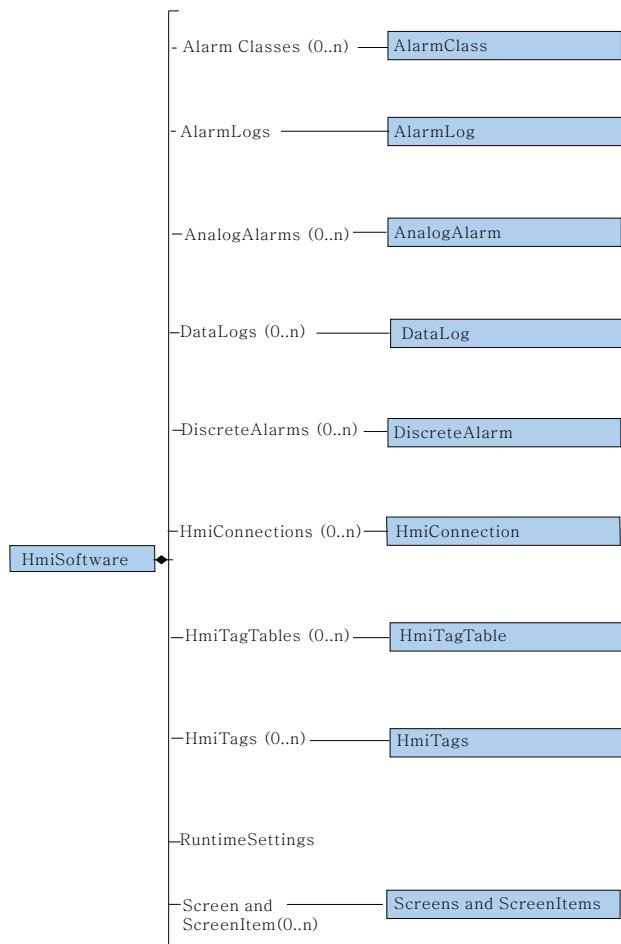


The following diagram describes the objects which are located under HmiTarget.



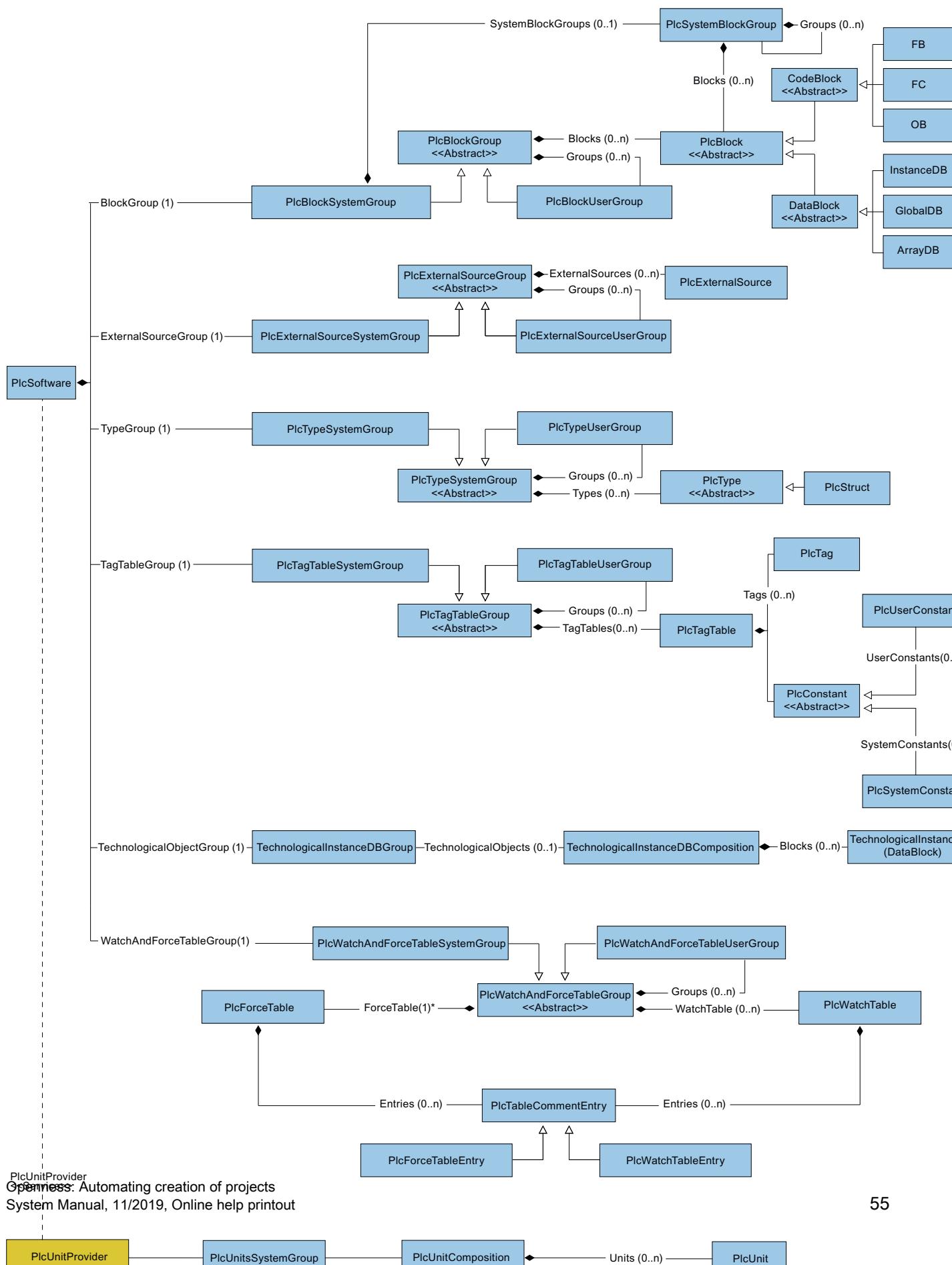
The following diagram describes the object which are located under HmiSoftware.

7.3 TIA Portal Openness object model



The following diagram describes the objects which are located under PlcSoftware.

7.3 TIA Portal Openness object model



7.3 TIA Portal Openness object model

Note

* The Force Table must be in PlcWatch and ForceTableSystemGroup

Access to objects in lists

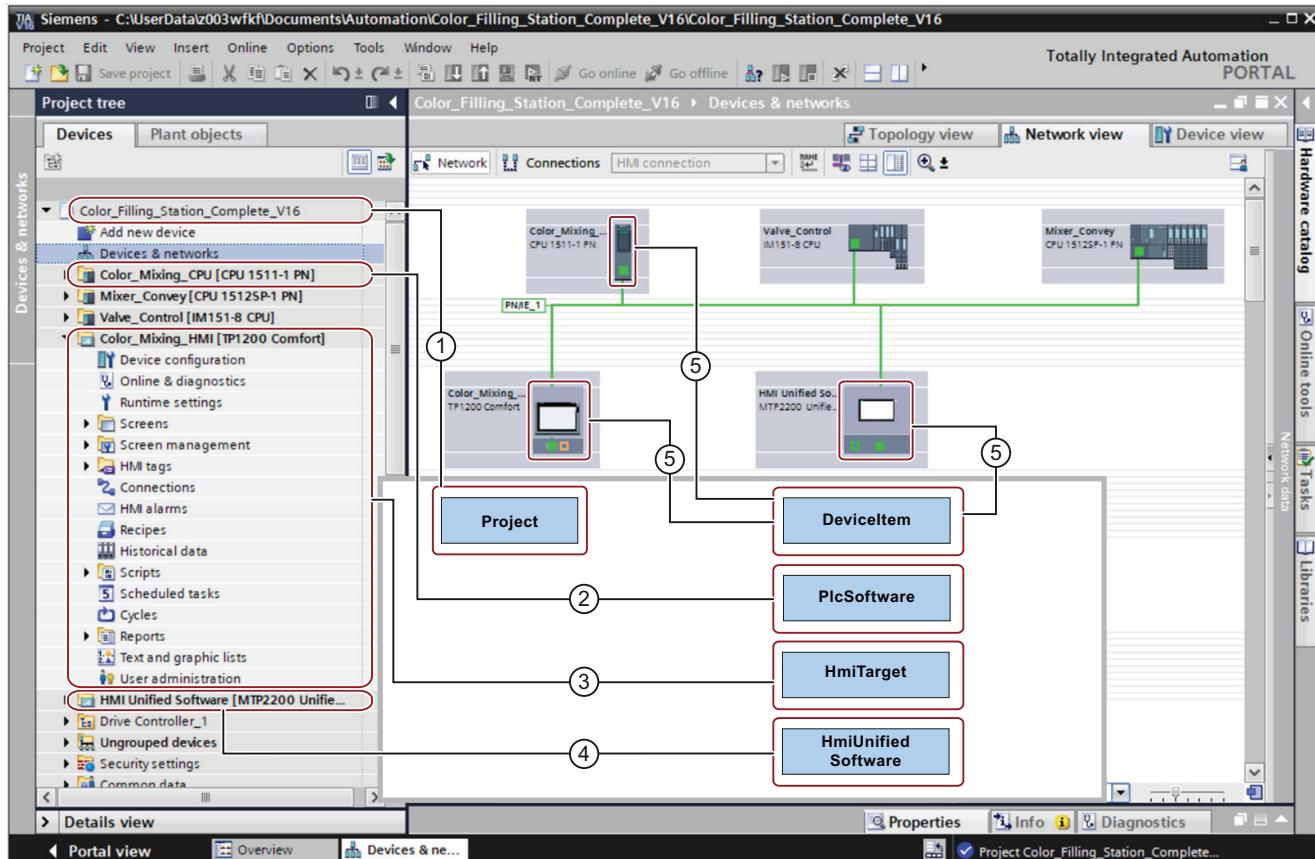
You have the following options for addressing an object in a list:

- Address via the index. The counting within the lists starts with 0.
- Use Find method.
Use this method to address an object via its name. You can use this method for an composition or list. The Find method is not recursive.
Example:

```
ScreenComposition screens = folder.Screens;
Screen screen = screens.Find("myScreen");
```
- Use symbolic names.

Relationship between TIA Portal and TIA Portal Openness object model

The figure below shows the relationship between the object model and a project in the TIA Portal:

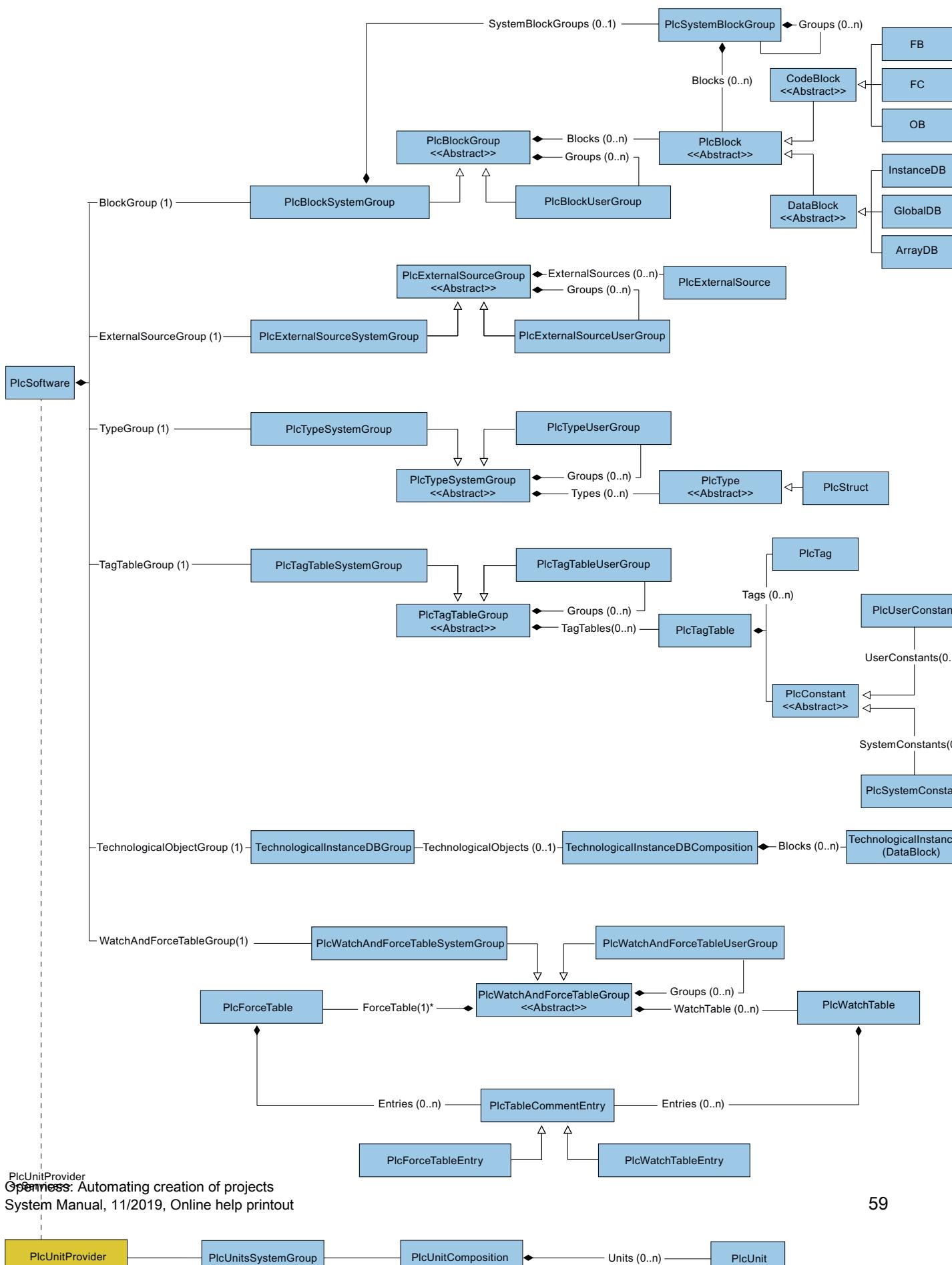


7.4 Blocks and types of the TIA Portal Openness object model

Introduction

The following diagram describes the domain model of the PLCs to give an overview of the current modeling in TIA Portal Openness.

7.4 Blocks and types of the TIA Portal Openness object model



Note

- * The Force Table must be in PlcWatch and ForceTableSystemGroup
-

Representation of blocks and types in the TIA Portal Openness API

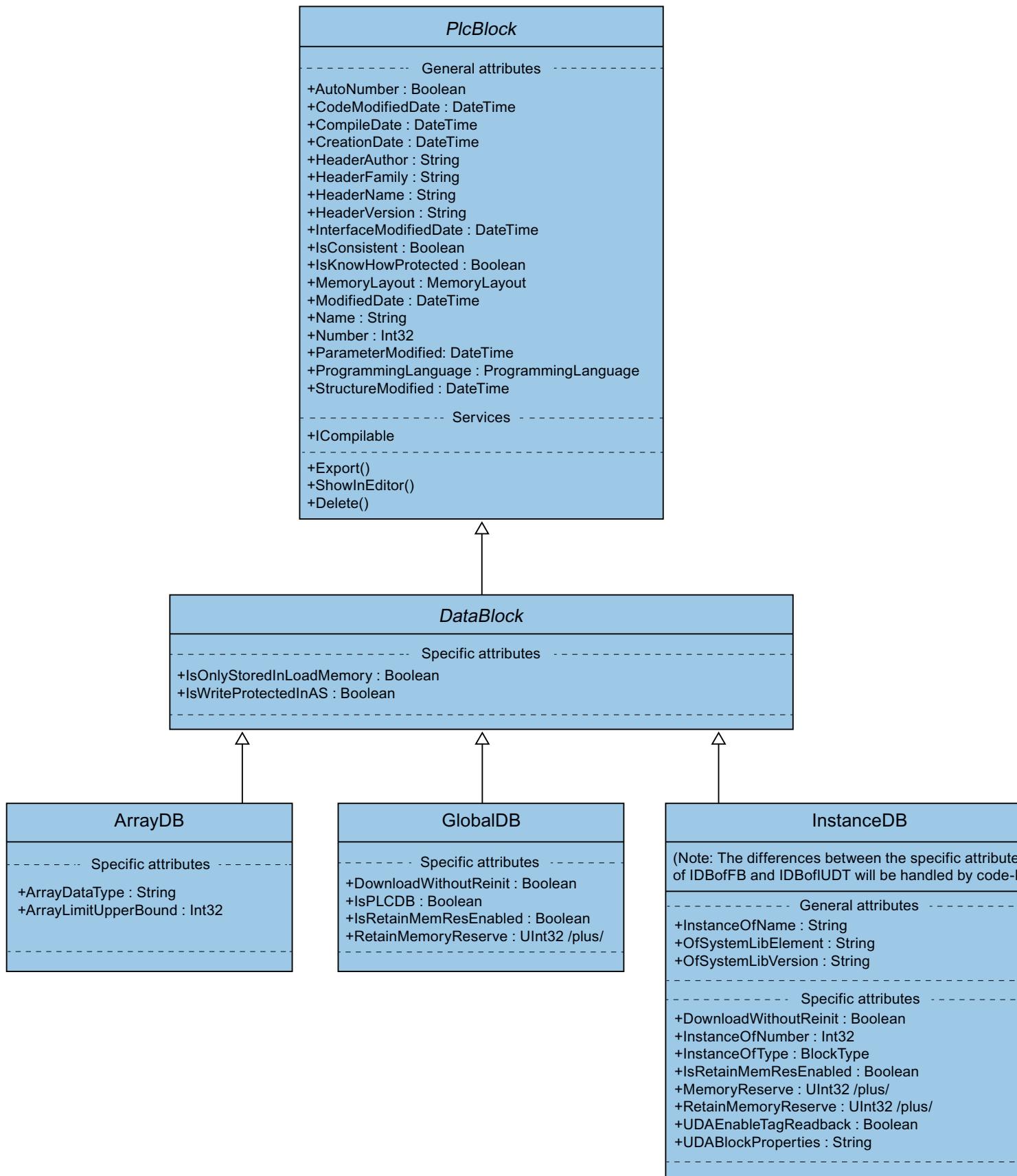
The simplified model part of blocks and for the structure is based on the attributes in the TIA Portal Openness API. These classes provide the export function and for blocks also the compile function.

Class diagrams

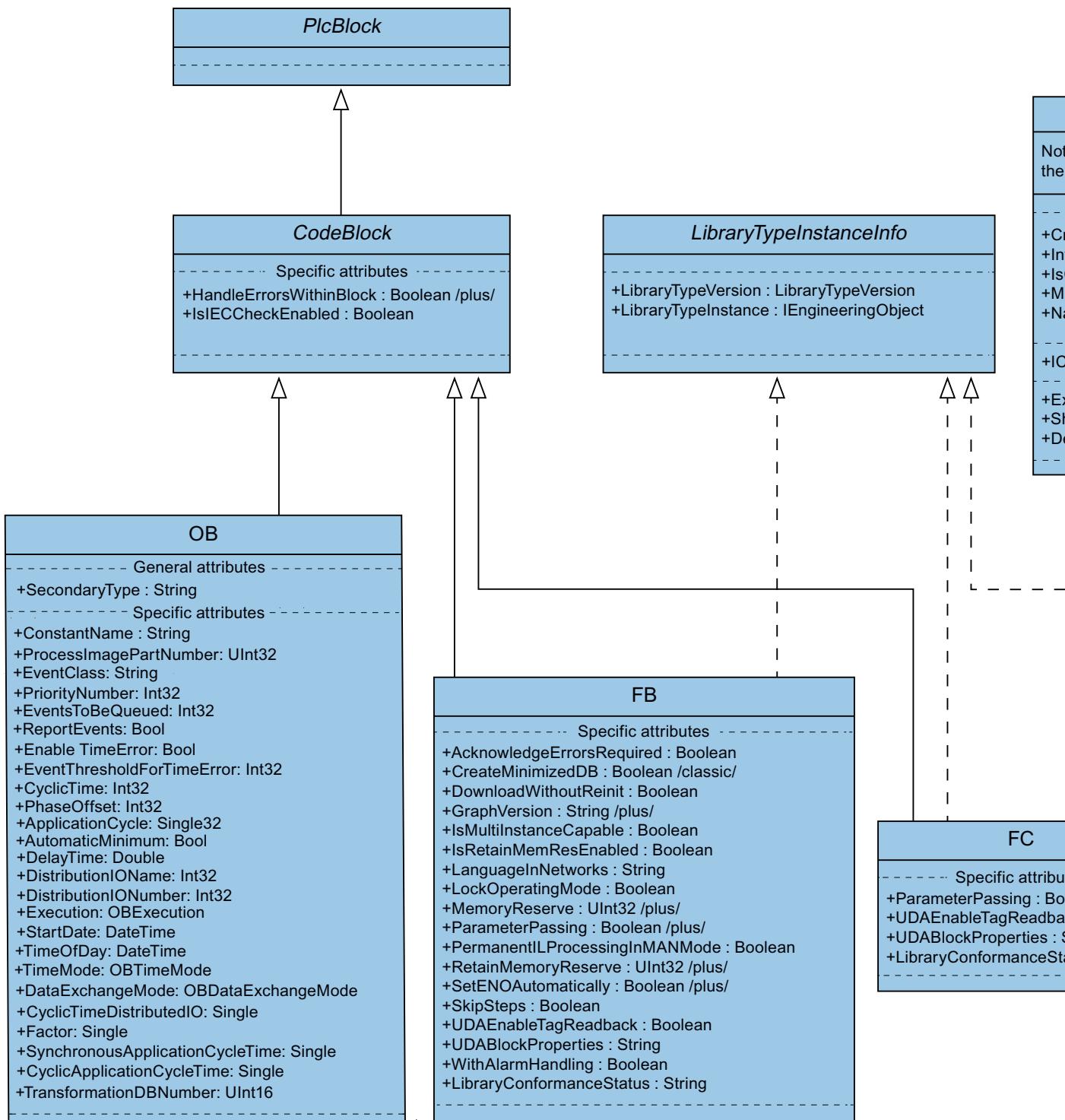
In the TIA Portal Openness object model all classes are defined as abstract which aren't directly instantiated.

Data

7.4 Blocks and types of the TIA Portal Openness object model



Code and Type

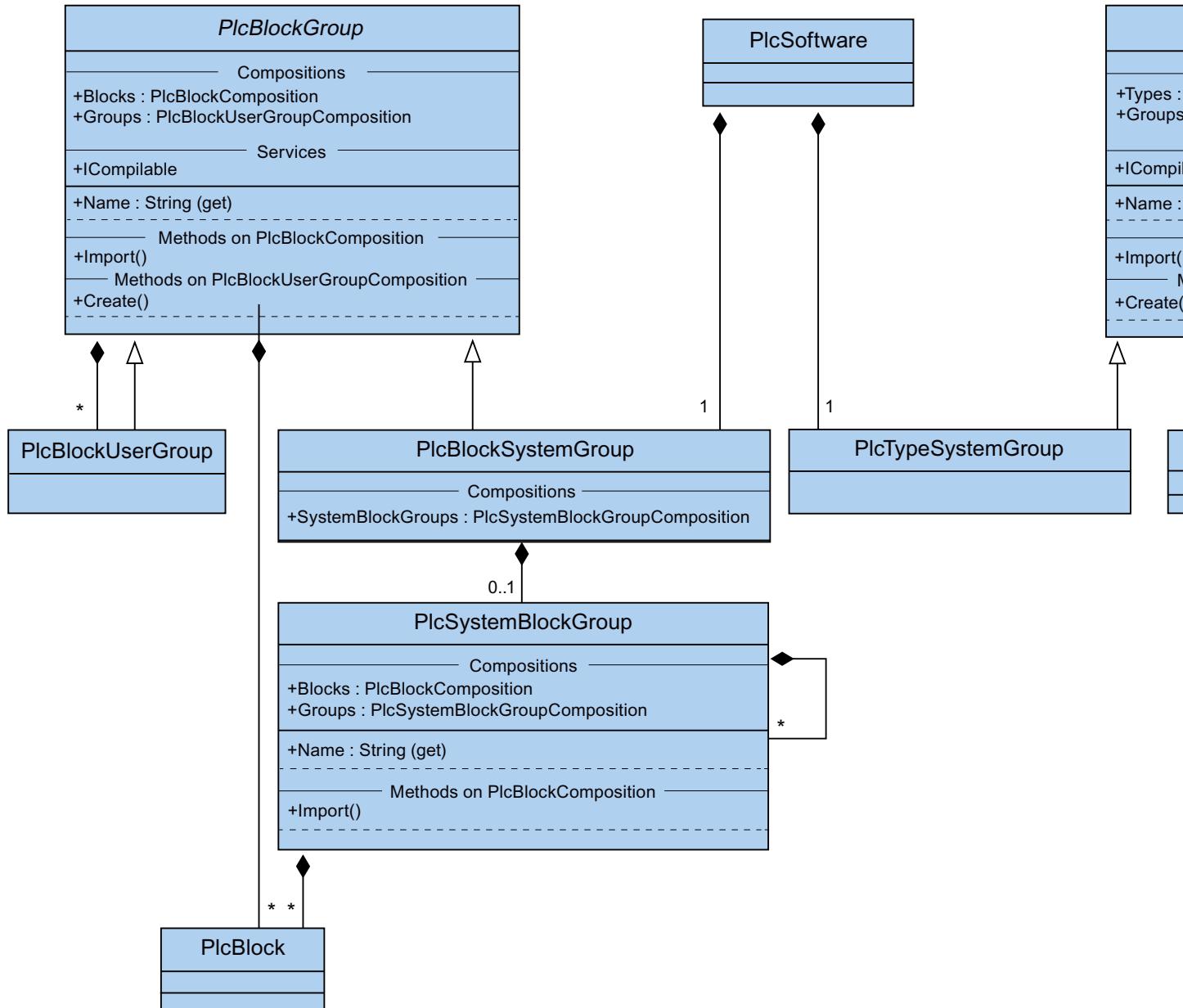


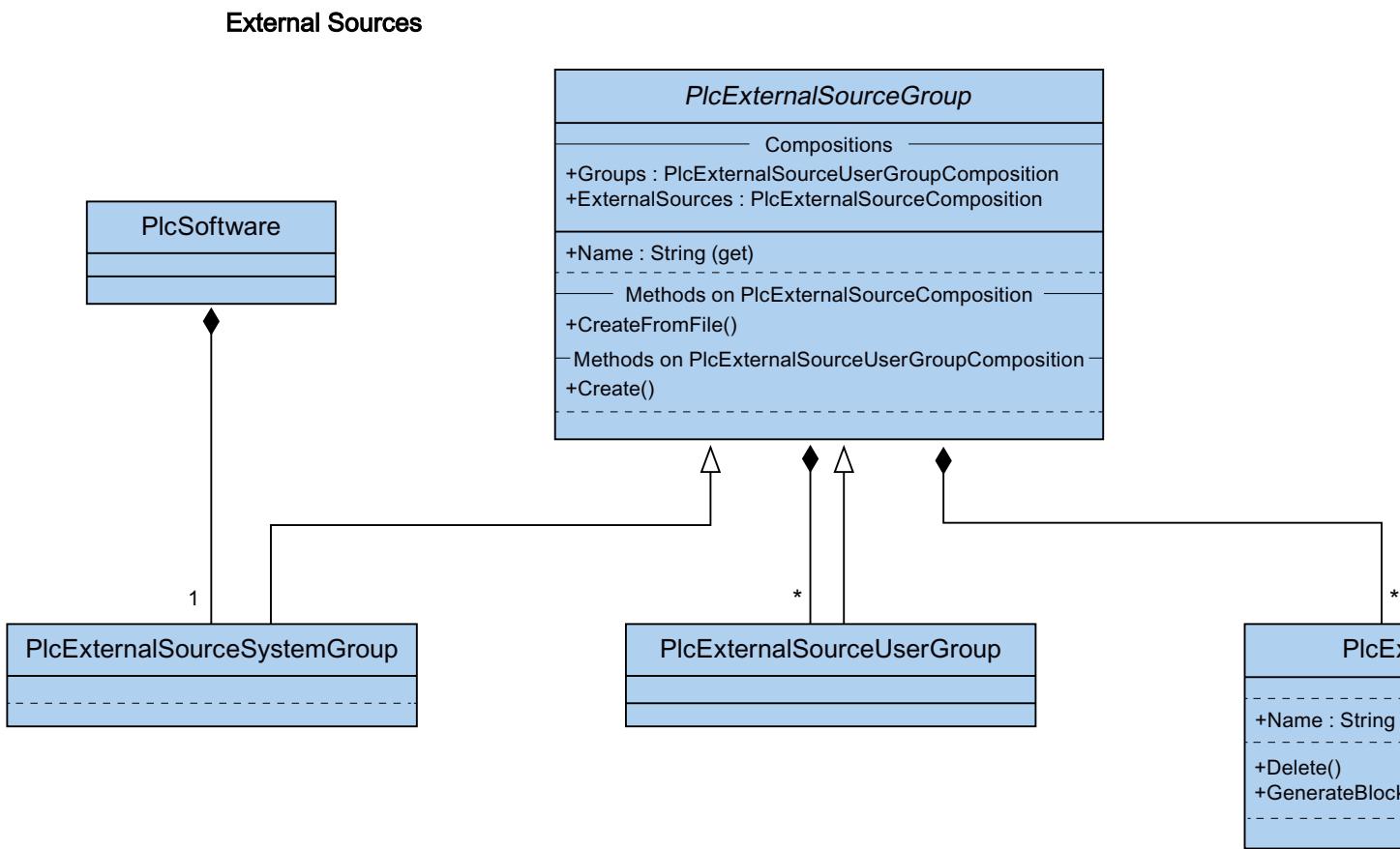
7.4 Blocks and types of the TIA Portal Openness object model

Representation of groups for blocks and types in the TIA Portal Openness API

The two high level groups "PlcBlocks" ("Program blocks" in the GUI of TIA Portal) and "PlcTypes" ("Plc data types" in the GUI of TIA Portal) contain blocks and type definitions. These groups provide the import and the compile functions for blocks. Due to the fact that most of the methods of functionalities of the groups are only achievable via collections, there is an "embedded" or "compacted" representation of the collections and their methods at the "host" classes.

Blocks and Types





7.5 Hierarchy of hardware objects of the object model

Relation between the visible elements in the TIA Portal and the modeled elements in the object model

Hardware object	Explanation
Device (Device)	The container object for a central or distributed configuration.
Device item (DeviceItem)	Each device item object has a container object. The logical relation is "Items".

The container relation is comparable to the relation of the modules for the device item objects.

Example: A device includes one or more slots. A slot includes modules. A module includes submodules.

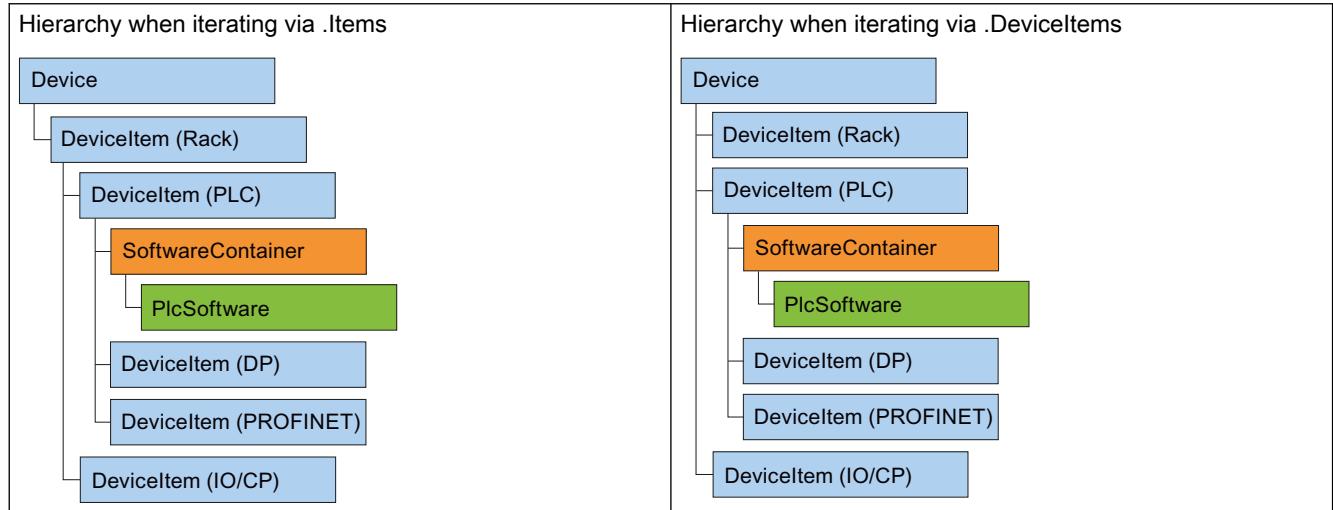
This is the relation similar to the representation in the network view and device view of the TIA Portal. The "PositionNumber" attribute of a device item is unique in the items area, within a container.

The parent-child relation between device item objects is a purely logical relation in the object model. A child cannot exist without its parents.

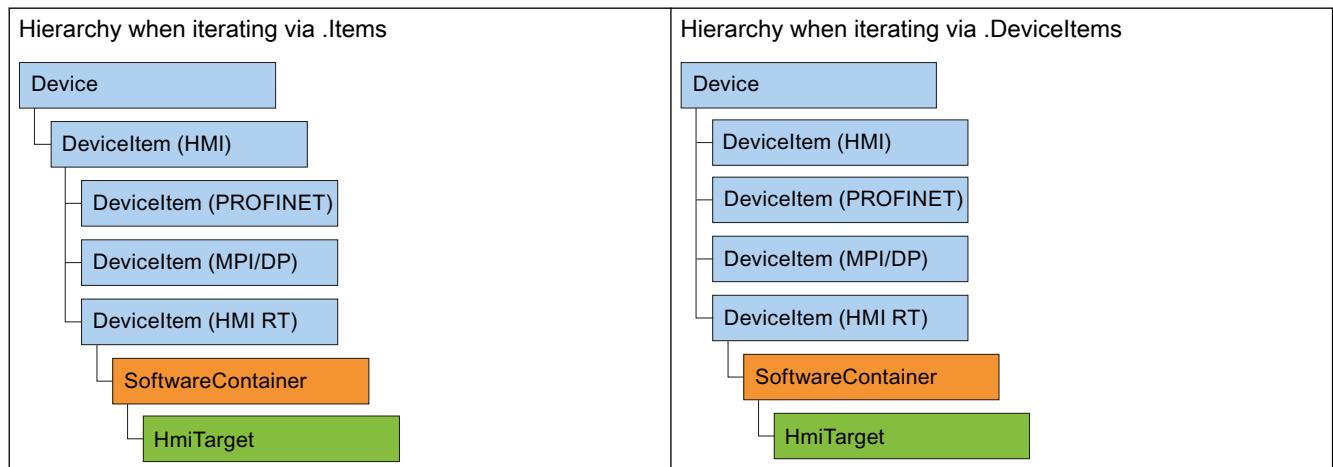
- If a submodule is modeled as part of a module (child), the submodule cannot be removed without the module.
- If you add and then remove a submodule from the module, this child has the same parents as the module.

The diagrams below show the hierarchy relationship between devices, and device items of PLC and HMI devices.

Hierarchy relationships of PLC devices



Hierarchy relationships of HMI devices



7.6 Information about installed TIA Portal Openness versions

Requirement

- TIA Portal Openness and TIA Portal are installed

Application

Starting from TIA Portal Openness V14 each installed version has a registry key that contains information about the version. This enables an automatic generation of app.config files for each installed version of TIA Portal Openness.

The registry keys can be located under the following path:

HKEY_LOCAL_MACHINE\Software\Siemens\Automation\Openness\Vxx.x
\PublicAPI

Note

The version number in this path, is always the number of the currently installed version of TIA Portal. If there are multiple side-by-side installations there are multiple sets of entries for TIA Portal Openness in the registry.

There is a single key for each version of TIA Portal Openness. The names of the Versions will be the same as in the assembly described, for example, the registry entries for TIA Portal Openness:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\Automation\Openness\Vxx.x\PublicAPI  
\Vxx.x.x.x]"PublicKeyToken"="d29ec89bac048f84"  
"Siemens.Engineering"="C:\Program Files\Siemens\Automation\Portal Vxx\PublicAPI\Vxx  
\Siemens.Engineering.dll"  
"Siemens.Engineering.Hmi"="C:\Program Files\Siemens\Automation\Portal Vxx\PublicAPI\Vxx  
\Siemens.Engineering.Hmi.dll"  
"EngineeringVersion"="Vxx"  
"AssemblyVersion"="Vxx.x.x.x"
```

Note

If you want to generate an app.config file you can get the path of the Siemens.Engineering.dll, the Siemens.Engineering.Hmi.dll and the public key token from the registry key.

7.7 Application example: Creating API access in a Windows Forms application

Application example: Creating API access in an application

The complete program code of the application example is shown below. The typical programming steps are explained next based on this example.

Note

The application example requires an application configuration file.

7.7 Application example: Creating API access in a Windows Forms application

```

using System;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.ExternalSources;
using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
using HmiTarget = Siemens.Engineering.Hmi.HmiTarget;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Library;
using System.IO;
namespace HelloTIA
{
    internal class Program
    {
        private static void Main(string[] args)
        {
            RunTiaPortal();
        }

        private static void RunTiaPortal()
        {
            Console.WriteLine("Starting TIA Portal");
            using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface))
            {
                Console.WriteLine("TIA Portal has started");
                ProjectComposition projects = tiaPortal.Projects;

                Console.WriteLine("Opening Project...");
                // please adapt the path and the extension apx to the installed version of
TIA Portal
                FileInfo projectPath = new FileInfo("C:\\Demo\\AnyCompanyProject.apx"); //edit
the path according to your project
                Project project = null;
                try
                {
                    project = projects.OpenWithUpgrade(projectPath);
                }
                catch (Exception)
                {
                    Console.WriteLine(String.Format("Could not open project {0}", projectPath.FullName));
                    Console.WriteLine("Demo complete hit enter to exit");
                    Console.ReadLine();
                }
            }
        }
    }
}

```

```
        return;
    }

    Console.WriteLine(String.Format("Project {0} is open",
project.Path.FullName));

    IterateThroughDevices(project);

    project.Close();

    Console.WriteLine("Demo complete hit enter to exit");
    Console.ReadLine();
}

}

private static void IterateThroughDevices(Project project)
{
    if (project == null)
    {
        Console.WriteLine("Project cannot be null");
        return;
    }

    Console.WriteLine(String.Format("Iterate through {0} device(s)",
project.Devices.Count));

    foreach (Device device in project.Devices)
    {
        Console.WriteLine(String.Format("Device: \"{0}\".", device.Name));
    }

    Console.WriteLine();
}
}
```

Procedure in steps

1. Make the TIA Portal known in the development environment

In your development environment, create a reference to all "dll files" in the "C:\Program Files \Siemens\Automation\PortalV..\\PublicAPI\\V..." directory.

The following provides a description of this process using the "Siemens.Engineering.dll" file as an example.

The "Siemens.Engineering.dll" file is available in the directory "C:\Program Files\Siemens\Automation\PortalV..\\PublicAPI\V..". Create a reference to the "Siemens.Engineering.dll" file in your development environment.

Note

Ensure that parameter "CopyLocal" is assigned the value "False" in the reference attributes.

2. Publish the name space for the TIA Portal

Add the following code:

```
using Siemens.Engineering;
```

3. Publish and start the TIA Portal

In order to publish and start the TIA Portal, insert the following code:

```
using (TiaPortal tiaPortal = new TiaPortal())
{
    // Add your code here
}
```

4. Open project

You can use the following code, for example, to open a project:

```
ProjectComposition projects = tiaPortal.Projects;
Console.WriteLine("Opening Project...");
//please adapt the path and the extension apx to the installed version of TIA Portal
FileInfo projectPath = new FileInfo("C:\Demo\AnyCompanyProject.apx");
Project project = null;
try
{
    project = projects.Open(projectPath);
}
catch (Exception)
{
    Console.WriteLine(String.Format("Could not open project {0}", projectPath.FullName));
    Console.WriteLine("Demo complete hit enter to exit");
    Console.ReadLine();
    return;
}
Console.WriteLine(String.Format("Project {0} is open", project.Path.FullName));
```

5. Enumerate devices of a project

Insert the following code to enumerate all devices of the project:

```
static private void IterateThroughDevices(Project project)
{
    if (project == null)
    {
        Console.WriteLine("Project cannot be null");
        return;
    }

    Console.WriteLine();
    Console.WriteLine(String.Format("Iterate through {0} device(s)", project.Devices.Count));
    foreach (Device device in project.Devices)
    {
        Console.WriteLine(String.Format("Device: \"{0}\".", device.Name));
    }
    Console.WriteLine();
}
```

6. Save and close the project

Insert the following code to save and close the project:

```
project.Save();
project.Close();
```

7.8 Use of the code examples

Structure of the code-snippets

Each code-snippet in this documentation is implemented as a function without return value with an object reference as transfer parameter. Disposing of objects is omitted for the sake of readability. Objects of the TIA Portal are addressed by their name using the `Find` method.

```
//Deletes a single screen from a user folder or a system folder
private static void DeleteScreenFromFolder(HmiTarget hmiTarget)
{
    //The screen "MyScreen" will be deleted if it is existing in the folder
    "myScreenFolder".
    //If "myScreen" is stored in a subfolder of "myScreenFolder" it will not be deleted.
    string screenName = "MyScreen";
    ScreenUserFolder folder = hmiTarget.ScreenFolder.Folders.Find("myScreenFolder");
    ScreenComposition screens = folder.Screens;
    Screen screen = screens.Find(screenName);
    if (screen != null)
    {
        screen.Delete();
    }
}
```

You need the following to execute this code-snippet:

- A WinCC project with an HMI device that includes a group with at least one screen.
- A function that instantiates the HMI device.

Note

When you specify directory paths, use the absolute directory path, for example, "C:/path/file.txt".

Relative directory paths are only allowed in the XML files for import and export, for example, "file.txt" or "C:/path01/.../path02/file.txt".

Example for execution of the code-snippet

Use the following example to execute the code-snippet "DeleteScreenFromFolder" as part of the "Hello TIA" example program:

```
//In the sample program "Hello TIA" replace the function call
//IterateThroughDevices(project) by the following functions calls:
HmiTarget hmiTarget = GetTheFirstHmiTarget(project);
DeleteScreenFromFolder(hmiTarget);

//Put the following function definitions before or after the
//function definition of "private static void IterateThroughDevices(Project project)":
private static HmiTarget GetTheFirstHmiTarget(Project project)
{
    if (project == null)
    {
        Console.WriteLine("Project cannot be null");
        throw new ArgumentNullException("project");
    }
    foreach (Device device in project.Devices)
        //This example looks for devices located directly in the project.
        //Devices which are stored in a subfolder of the project will not be affected by this
example.
    {
        foreach (DeviceItem deviceItem in device.DeviceItems)
        {
            DeviceItem deviceItemToGetService = deviceItem as DeviceItem;
            SoftwareContainer container =
deviceItemToGetService.GetService<SoftwareContainer>();
            if (container != null)
            {
                HmiTarget hmi = container.Software as HmiTarget;
                if (hmi != null)
                {
                    return hmi;
                }
            }
        }
    }
    return null;
}

//Deletes a single screen from a user folder or a system folder
private static void DeleteScreenFromFolder(HmiTarget hmiTarget)
{
    string screenName = "MyScreen";
    ScreenUserFolder folder = hmiTarget.ScreenFolder.Folders.Find("myScreenFolder");
    ScreenComposition screens = folder.Screens;
    Screen screen = screens.Find(screenName);
    if (screen != null)
    {
        screen.Delete();
    }
}
```

7.9 General functions

7.9.1 TIA Portal Openness IntelliSense support

Application

The Intellisense support of TIA Portal Openness helps you at available attributes or methods via tooltip information. It could contain information about the number, names and types of the required parameters. At the following example the bold parameter in the first line indicates the next parameter that is required as you type the function.

```
project = tiaPortal.Projects.OpenWithUpgrade(projectInfo);
```

Project ProjectComposition.OpenWithUpgrade(FileInfo path)
Open Action with project update is necessary

You can manually invoke Parameter Info by clicking Edit IntelliSense/Parameter Info, typing CTRL+SHIFT+SPACE, or clicking the Parameter Info button on the editor toolbar.

7.9.2 Connecting to the TIA Portal

Introduction

You start the TIA Portal with TIA Portal Openness or connect to a TIA Portal already running. When using a TIA Portal Openness application to start the TIA Portal, you specify if the TIA Portal should be started with or without graphical user interface. When you operate the TIA Portal without user interface, the TIA Portal is only started as a process by the operating system. You create several instances of the TIA Portal with a TIA Portal Openness application, if necessary.

Note

If you use TIA Portal Openness with the TIA Portal interface, you cannot use an HMI editor. You can open the "Devices & Networks" editors or the programming editor manually or with TIA Portal Openness API.

You have the following options to start the TIA Portal with a TIA Portal Openness application:

- Use an application configuration file (recommended in most use cases).
- Use the "AssemblyResolve" method (recommended when you use copy deploy etc.).
- Copy the Siemens.Engineering.dll in the TIA Portal Openness application directory.

Note

It is recommended to load the Siemens.Engineering.dll by using the app.config file. By using this method the strong names are considered and malicious modifications to the engineering.dll will result in a loading error. By using the AssemblyResolve method this can't be detected.

Starting the TIA Portal with an application configuration file

Reference all required program libraries in the application configuration file. You distribute the application configuration file together with the TIA Portal Openness application.

Store the application configuration file "app.config" in the same directory as the TIA Portal Openness application and likewise incorporate this in your application. Check whether the file path in each code matches the TIA Portal installation path.

You can use the following code snippet for the application configuration file:

```
<?xml version="1.0"?>
<configuration>
    <runtime>
        <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
            <dependentAssembly>
                <assemblyIdentity name="Siemens.Engineering" culture="neutral"
publicKeyToken="d29ec89bac048f84"/>
                <!-- Edit the following path according to your installed version of TIA Portal
-->
                <codeBase version="Vxx.x.x.x" href="FILE://C:\Program Files\Siemens
\Automation\Portal Vxx\PublicAPI\Vxx\Siemens.Engineering.dll"/>
            </dependentAssembly>
            <dependentAssembly>
                <assemblyIdentity name="Siemens.Engineering.Hmi" culture="neutral"
publicKeyToken="d29ec89bac048f84"/>
                <!-- Edit the following path according to your installed version of TIA Portal
-->
                <codeBase version="Vxx.x.x.x" href="FILE://C:\Program Files\Siemens
\Automation\Portal Vxx\PublicAPI\Vxx\Siemens.Engineering.Hmi.dll"/>
            </dependentAssembly>
        </assemblyBinding>
    </runtime>
</configuration>
```

Use the following program code to open a new TIA Portal Instance by means of the application configuration file:

```
//Connect a TIA Portal Openness application via API using
using System;
using System.IO;
using Siemens.Engineering;

namespace UserProgram
{
    internal class MyProgram
    {
        public static void Main(string[] args)
        {
            // To start TIA Portal with user interface:
            // using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface))
            //
            // To start TIA Portal without user interface:
            // using (TiaPortal tiaPortal = new
TiaPortal(TiaPortalMode.WithoutUserInterface))
            using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface))
            {
                //begin of code for further implementation
                //...
                //end of code
            }
        }
    }
}
```

Starting the TIA Portal using the "AssemblyResolve" method

Design the program code of the TIA Portal Openness application in such a way that you register on the event "AssemblyResolve" as early as possible. Encapsulate the access to the TIA Portal in an additional object or method.

Caution must be taken when resolving the engineering assembly using an assembly resolver method. If any types from the engineering assembly are used before the assembly resolver has had run, the program will crash. The reason for this is that the Just-in-time compiler (JIT compiler) doesn't compile methods until it needs to execute them. If engineering assembly types are used in Main, for example, the JIT compiler will attempt to compile Main when the program runs and fail because it doesn't know where to find the engineering assembly. The registration of the assembly resolver in Main doesn't change this. The method needs to run before the assembly resolver is registered, and it needs to be compiled before it can be run. The solution for this problem, is to place the business logic that uses types from the engineering assembly into a separate method that uses only types that the JIT compiler already understands. In the example, a method that returns void and has no parameters and place all business logic inside it is used. When the JIT compiler compiles Main, it will succeed because it knows all the types in Main. At runtime, when we call RunTiaPortal, the assembly resolver will already be registered, so when the JIT compiler tries to find our business logic types, it will know where to find the engineering assembly.

Use the following program code to open a new TIA Portal Instance.

```

using System;
using System.IO;
using System.Reflection;
using Siemens.Engineering;

namespace UserProgram
{
    static class MyProgram
    {
        public static void Main(string[] args)
        {
            AppDomain.CurrentDomain.AssemblyResolve += MyResolver;
            RunTiaPortal();
        }

        private static void RunTiaPortal()
        {
            // To start TIA Portal with user interface:
            // using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface))
            //
            // To start TIA Portal without user interface:
            // using (TiaPortal tiaPortal = new
TiaPortal(TiaPortalMode.WithoutUserInterface))
            using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface))
            {
                //begin of code for further implementation
                //...
                //end of code
            }
        }

        private static Assembly MyResolver(object sender, ResolveEventArgs args)
        {
            int index = args.Name.IndexOf(',');
            if (index == -1)
            {
                return null;
            }
            string name = args.Name.Substring(0, index) + ".dll";
            // Edit the following path according to your installed version of TIA Portal
            string path = Path.Combine(@"C:\Program Files\Siemens\Automation\Portal Vxx
\PublicAPI\Vxx\", name);
            string fullPath = Path.GetFullPath(path);
            if (File.Exists(fullPath))
            {
                return Assembly.LoadFrom(fullPath);
            }
            return null;
        }
    }
}

```

Accessing running instances of the TIA Portal

In order to connect to a running instance of the TIA Portal with a TIA Portal Openness application, start by enumerating the instances of the TIA Portal. You can connect to multiple instances within a Windows session. The running instance can be TIA Portal with or without a started user interface:

```
foreach (TiaPortalProcess tiaPortalProcess in TiaPortal.GetProcesses())
{
    //...
}
```

If you know the process ID of the instance of the TIA Portal, use this process ID to access the object. TIA Portal requires a certain amount of time to start up before you can connect the TIA Portal Openness application to the TIA Portal.

When you connect to a running instance of the TIA Portal, a connection prompt of the TIA Portal Openness firewall appears. The connection prompt offers the following options:

- Allow connection once
 - Do not allow connection
 - Always allow connections from this application
- See TIA Portal Openness firewall (Page 81) for further information.

Note

If the registry prompt is rejected three times, the system throws an exception of the type `EngineeringSecurityException`.

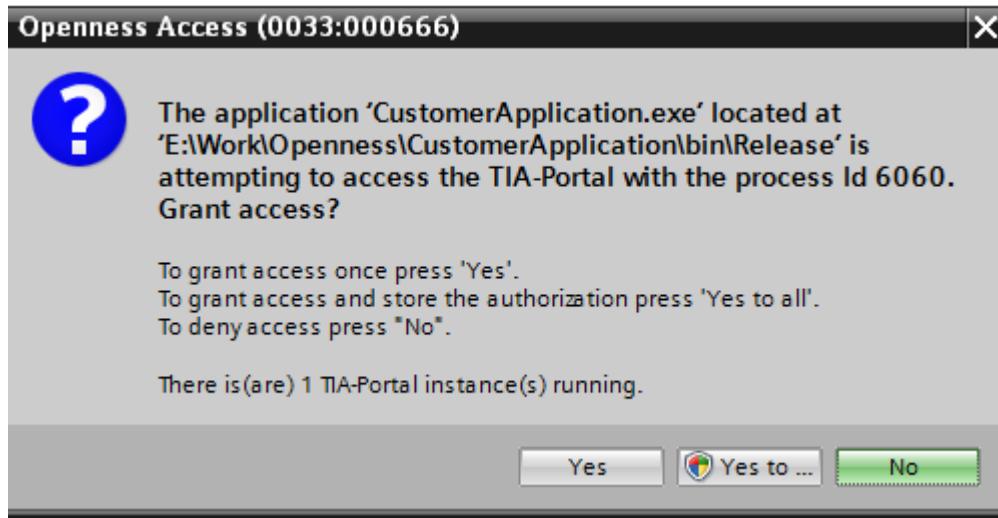
Once you have connected to the process, you can use the following attributes to retrieve information on the instances of the TIA Portal:

Attribute	Information
InstalledSoftware as IList<TiaPortalProduct>	Returns information about the installed products.
Mode as TiaPortalMode	Returns the mode in which the TIA Portal was started (WithoutUserInterface/WithUserInterface).
AttachedSessions as IList<TiaPortalSession>	Returns a list of applications connected to the TIA Portal.
ProjectPath as FileInfo	Returns the file name of the project opened in the TIA Portal, including the folder, for example, "D:\WinCCProjects\ColorMixing\ColorMixing.ap*" If no project is open, a null string is returned.
ID as int	Returns the process ID of the TIA Portal instance
Path as FileInfo	Returns the path to the TIA Portal executable

7.9.3 TIA Portal Openness firewall

TIA Portal Openness firewall prompt

When you try to connect to a running TIA Portal via TIA Portal Openness, the TIA Portal will prompt you to accept or reject the connection like the following screenshot is showing.



Allow connection to the TIA Portal once

If you just want to connect your TIA Portal Openness application to the TIA Portal once, click "Yes" at the prompt. The next time your TIA Portal Openness application tries to connect the TIA Portal, the prompt will be shown again.

Addition of a whitelist entry by connecting the TIA Portal

To create a whitelist entry for your TIA Portal Openness application follow these steps:

1. Click "Yes to all" at the prompt to display an User Account Control Dialog.
2. Click "Yes" at the User Account Control Dialog to add your application to the whitelist in the windows registry and to attach the application to the TIA Portal.

Addition of a whitelist entry without using the TIA Portal

If you want to add an entry to the whitelist without using TIA Portal you can create a reg file like this:

```
Windows Registry Editor Version 5.00
[HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\Automation\Openness\Vxx.x\Whitelist
\CustomerApplication.exe]
[HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\Automation\Openness\Vxx.x\Whitelist
\CustomerApplication.exe\Entry]
"Path"="E:\\Work\\Openness\\CustomerApplication\\bin\\Release\\CustomerApplication.exe"
"DateModified"="2014/06/10 15:09:44.406"
"FileHash"="0rXRKUCNzMWHOMFrT52OwXzqJef10ran4UykTeBraaY="
```

The following example shows how you can calculate the file hash and last modified date:

```
string applicationPath = @"E:\\Work\\Openness\\CustomerApplication\\bin\\Release\\CustomerApplication.exe";
string lastWriteTimeUtcFormatted = String.Empty;
DateTime lastWriteTimeUtc;
HashAlgorithm hashAlgorithm = SHA256.Create();
FileStream stream = File.OpenRead(applicationPath);
byte[] hash = hashAlgorithm.ComputeHash(stream);
// this is how the hash should appear in the .reg file
string convertedHash = Convert.ToBase64String(hash);
lastWriteTimeUtc = fileInfo.LastWriteTimeUtc;
// this is how the last write time should be formatted
lastWriteTimeUtcFormatted = lastWriteTimeUtc.ToString(@"yyyy/MM/dd HH:mm:ss.ffff");
```

7.9.4 Event handlers

Event handlers in TIA Portal Openness application

An instance of the TIA Portal provides the following events to which you can react with an event handler in a TIA Portal Openness application. You can access the attributes of notifications and define the responses accordingly.

Event	Response
Disposed	Use this event to respond to the closing of the TIA Portal with a TIA Portal Openness application.
Notification	Use this event to respond to notifications of the TIA Portal with a TIA Portal Openness application. Notifications require only an acknowledgment, e.g. "OK".
Confirmation	Use this event to respond to confirmations of the TIA Portal with a TIA Portal Openness application. Confirmations always require a decision, e.g. "Do you want to save the project?".

Program code

Modify the following program code to register event handlers in a TIA Portal Openness application:

```
//Register event handler for Disposed-Event
.....
    tiaPortal.Disposed += TiaPortal_Disposed;
.....
private static void TiaPortal_Disposed(object sender, EventArgs e)
{
    .....
}

//Register event handler for Notification-Event
.....
    tiaPortal.Notification += TiaPortal_Notification;
.....
private static void TiaPortal_Notification(object sender, NotificationEventArgs e)
{
    .....
}

//Register event handler for Confirmation-Event
.....
    tiaPortal.Confirmation += TiaPortal_Confirmation;
.....
private static void TiaPortal_Confirmation(object sender, ConfirmationEventArgs e)
{
    .....
}
```

Attributes of TIA Portal notifications

TIA Portal notifications have the following attributes:

Attribute	Description
Caption	Returns the name of the confirmation.
DetailText	Returns the detail text of the confirmation.
Icon	Returns the icon of the confirmation.
IsHandled	Returns the confirmation or specifies if it is still pending.
Text	Returns the text of the confirmation.

Attributes of confirmations

Confirmations have the following attributes:

Attribute	Description
Caption	Returns the name of the confirmation.
Choices	Returns the option to acknowledge the confirmation.
DetailText	Returns the detail text of the confirmation.
Icon	Returns the icon of the confirmation.
IsHandled	Returns the confirmation or specifies if it is still pending.
Result	Returns the result of the acknowledgment or specifies it.
Text	Returns the text of the confirmation.

See also

[Program-controlled acknowledgement of dialogs with system events \(Page 84\)](#)

7.9.5 Program-controlled acknowledgement of dialogs with system events

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
[See Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
[See Opening a project \(Page 106\)](#)
- Event handlers are registered.
[See Connecting to the TIA Portal \(Page 76\)](#)

Application

When you operate the TIA Portal with the user interface, dialogs with system events are displayed for some program sequences. You decide how you want to proceed based on these system events.

When the TIA Portal is accessed with a TIA Portal Openness application, these system events must be acknowledged by means of corresponding ".NET" events.

The permitted confirmations are contained in the `Choices` list:

- Abort
- Cancel
- Ignore
- No
- NoToAll
- None

- OK
- Retry
- Yes
- YesToAll

The value of ConfirmationEventArgs.Result must be one of the above-mentioned entries. Otherwise, an exception is thrown.

Program code

Modify the following program code to respond to a confirmation event:

```
...
    tiaPortal.Confirmation += TiaPortalConfirmation;
...
private void TiaPortalConfirmation(object sender, ConfirmationEventArgs e)
{
    ...
}
```

Modify the following program code to notify the project engineer about executed actions of a TIA Portal Openness application:

```
//Handles notifications
using (TiaPortal tiaPortal = new TiaPortal())
{
    tiaPortal.Notification += Notification;
    try
    {
        //perform actions that will result in a notification event
    }
    finally
    {
        tiaPortal.Notification -= Notification;
    }
}
```

7.9.6 Terminating the connection to the TIA Portal

Introduction

If you started the TIA Portal instance without a user interface and if your application is the only TIA Portal Openness client attached to the TIA Portal, you can close the TIA Portal instance with the TIA Portal Openness application. Otherwise, you disconnect the TIA Portal Openness application from the TIA Portal instance.

Use the `IDisposable.Dispose()` method to separate or close the active instance of the TIA Portal.

You can use the `IDisposable.Dispose()` method as follows:

- With a using statement.
- Surround the object description with a try-finally block and call the `IDisposable.Dispose()` method within the finally block.

You can no longer access the TIA Portal when you close the active instance of the TIA Portal.

Note

When a configuration engineer closes the TIA Portal instance despite ongoing access of a TIA Portal Openness application, an exception of the class "NonRecoverableException" is thrown in the TIA Portal Openness application on the next API access. You can subscribe to the dispose event to get a call when the TIA Portal is closed.

Program code

Modify the following program code to separate or close the connection to the TIA Portal:

```
// Add code to dispose the application if the application is still instantiated
if (tiaPortal != null)
{
    tiaPortal.Dispose();
}
```

See also

[Event handlers \(Page 82\)](#)

7.9.7 Diagnostic interfaces on TIA Portal

Application

You can retrieve certain diagnostic information from running instances of TIA Portal via a static method. The diagnostic interface is implemented on the `TiaPortalProcess` object, which can be retrieved for any currently running instance of the TIA Portal.

The diagnostic interface is not blocking, so you can retrieve the `TiaPortalProcess` object and access its members, regardless of if the TIA Portal is busy or not. The diagnostic interface includes the following Members:

Class TiaPortalProcess

Member	Type	Function
AcquisitionTime	DateTime	The time when the TiaPortalProcess object was acquired. Since the TiaPortalProcess object represents a completely static snapshot of the state of the TIA Portal at a given point in time, the information it contains may become outdated.
Attach	TiaPortal	Attaches to the given TiaPortalProcess, it returns a TiaPortal instance.
AttachedSessions	IList<TiaPortalSession>	A collection of all other sessions currently attached to the same TIA Portal. This collection can be empty. Each session is represented by a TiaPortalSession object.
Attaching	EventHandler<AttachingEventArgs>	This event enables an application to approve any attempts to attach to the TIA Portal. When another application attempts to attach to the TIA Portal, the subscribers of this event are notified and given 10 seconds to approve the attachment. If any subscriber ignores this event or does not respond in time, it is understood to be denying the other application permission to attach. Crashed applications, being unable to respond to this event and cannot cause an application to be denied permission to attach.
Dispose	void	Closes the associated TIA Portal instance.
Id	int	The Process ID of the TIA Portal.
InstalledSoftware	IList<TiaPortalProduct>	A collection of all the products currently installed as part of the TIA Portal. Each product is represented by a TiaPortalProduct object, which is described below.
Mode	TiaPortalMode	The mode in which the TIA Portal was started. The current values are WithUserInterface and WithoutUserInterface.
Path	FileInfo	The path to the executable of the TIA Portal.
ProjectPath	FileInfo	The path to the project which is currently open in the TIA Portal. If no project is open, this attribute will be null.

Class TiaPortalSession

Member	Type	Function
AccessLevel	TiaPortalAccessLevel	The level access of the session. It is represented as a flags enum, where multiple levels of access are possible. TiaPortalAccessLevel is described in detail below.
AttachTime	DateTime	The time when the connection to the TIA Portal was established.
Id	int	The Id of the current session.
IsActive	bool	Returns "true" if the TIA Portal is currently processing a call from the running session.
Dispose	void	Severs the process's connection to the TIA Portal. This method does not kill the process itself in the way that System.Diagnostics.Process.Kill would do. The application whose connection is terminated will still get a disposed event, but there is no other indication of why the connection was terminated.
ProcessId	int	The process ID of the attached process.
ProcessPath	FileInfo	The path to the executable of the attached process.
TrustAuthority	TiaPortalTrustAuthority	Indicates if the current session was started by a process that was signed, and if it is a TIA Portal Openness certificate or not. TrustAuthority is a flags enum and is described below.
UtilizationTime	TimeSpan	The period of time the process has spent actively using the TIA Portal. Combined with the AttachTime attribute, this could be used to determine usage percentages or similar data.
Version	string	The version of the Siemens.Engineering.dll to which the session is attached to.

Enum TiaPortalAccessLevel

Enum Value	Function
None	This is not a valid value. It is included because TiaPortalAccessLevel is a flags enum which needs an appropriate "zero value" to represent no flags being set, but it will never appear in actual use because no session can be started that has no access.
Published	The session has access to published functionality.
Modify	The session has modify access.

Enum TiaPortalTrustAuthority

Enum Value	Function
None	The main module of the attached process is not signed with a certificate.
Signed	The main module is signed with a certificate, which is not a TIA Portal Openness certificate.
Certified	The main module is signed with a TIA Portal Openness certificate.
CertifiedWithExpiration	The main module is signed with a TIA Portal Openness certificate that will become invalid at the end of its lifetime.

Class TiaPortalProduct

Member	Type	Function
Name	string	The name of the product (e.g. STEP 7 Professional).
Options	IList<TiaPortalProduct>	A collection of all optional packages that belong to the connected TIA Portal, represented as TiaPortalProduct objects. If an option package itself has option packages, this nesting could continue.
Version	string	The version string of the product.

7.9 General functions

The following code snippet provides an example of how to use the diagnostic Interface to query information and of how to use them in your application.

```

public void TiaPortalDiagnostics()
{
    IList<TiaPortalProcess> tiaPortalProcesses = TiaPortal.GetProcesses();
    foreach (TiaPortalProcess tiaPortalProcess in tiaPortalProcesses)
    {
        Console.WriteLine("Process ID: {0}", tiaPortalProcess.Id);
        Console.WriteLine("Path: {0}", tiaPortalProcess.Path);
        Console.WriteLine("Project: {0}", tiaPortalProcess.ProjectPath);
        Console.WriteLine("Timestamp: {0}", tiaPortalProcess.AcquisitionTime);
        Console.WriteLine("UI Mode: {0}", tiaPortalProcess.Mode);
        //See method body below.
        Console.WriteLine("Installed Software:");
        EnumerateInstalledProducts(tiaPortalProcess.InstalledSoftware);
        Console.WriteLine("Attached Openness Applications:");
        foreach (TiaPortalSession session in tiaPortalProcess.AttachedSessions)
        {
            Console.WriteLine("Process: {0}", session.ProcessPath);
            Console.WriteLine("Process ID: {0}", session.ProcessId);
            DateTime attachTime = session.AttachTime;
            TimeSpan timeSpentAttached = DateTime.Now - attachTime;
            TimeSpan utilizationTime = session.UtilizationTime;
            long percentageTimeUsed = (utilizationTime.Ticks / timeSpentAttached.Ticks) *
100;
            Console.WriteLine("AttachTime: {0}", attachTime);
            Console.WriteLine("Utilization Time: {0}", utilizationTime);
            Console.WriteLine("Time spent attached: {0}", timeSpentAttached);
            Console.WriteLine("Percentage of attached time spent using TIA Portal: {0}",
percentageTimeUsed);
            Console.WriteLine("AccessLevel: {0}", session.AccessLevel);
            Console.WriteLine("TrustAuthority: {0}", session.TrustAuthority);
            if ((session.TrustAuthority & TiaPortalTrustAuthority.Certified) !=
TiaPortalTrustAuthority.Certified)
            {
                Console.WriteLine("TrustAuthority doesn't match required level, attempting
to terminate connection to TIA Portal.");
                session.Dispose();
            }
        }
    }
}
public void EnumerateInstalledProducts(IEnumerable<TiaPortalProduct> products)
{
    foreach (TiaPortalProduct product in products)
    {
        Console.WriteLine("Name: {0}", product.Name);
        Console.WriteLine("Version: {0}", product.Version);
        //recursively enumerate all option packages
        Console.WriteLine("Option Packages \n:");
        EnumerateInstalledProducts(product.Options);
    }
}

```

Security Relevant Information

Because of the fact that no connection to the TIA Portal is needed to use the diagnostics interface, it's possible to write a Windows service that uses the attaching event to check any application attempting to attach to a TIA Portal, e.g. only applications that begin with your company's name are allowed to attach. Another option might be to always grant access, but write information about attaching processes to a log. The following program code is an example event handler to check incoming connections:

```
public void OnAttaching(object sender, AttachingEventArgs e)
{
    string name = Path.GetFileNameWithoutExtension(e.ProcessPath);
    TiaPortalAccessLevel requestedAccessLevel = e.AccessLevel &
    TiaPortalAccessLevel.Published;
    TiaPortalTrustAuthority certificateStatus = e.TrustAuthority
    & TiaPortalTrustAuthority.Certified;
    if (requestedAccessLevel == TiaPortalAccessLevel.Published &&
        certificateStatus == TiaPortalTrustAuthority.Certified &&
        name.StartsWith("SampleCustomerName"))
    {
        e.GrantAccess();
    }
}
```

7.9.8 Exclusive access

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The "TIA Portal" class provides the method "ExclusiveAccess(String text)" to establish an exclusive access to an attached TIA Portal process. The usage of an exclusive access is highly recommended even if it is not mandatory.

Use "ExclusiveAccess" in an "using" statement to ensure it is disposed attribute even when exceptions occur or the application is shutdown.

Note

Any attempt to create a second exclusive access within the scope of an open exclusive access will result in an recoverable exception being raised.

Modify the following example to get "ExclusiveAccess" to an instance:

```
...
[assembly: AssemblyTitle("MyApplication")]
// This will be used for the exclusive access dialog when present....
TiaPortal tiaPortal = ...;
using (ExclusiveAccess exclusiveAccess = tiaPortal.ExclusiveAccess("My Activity"))
{
    ...
}
```

After acquiring an "ExclusiveAccess" instance for a given TIA Portal process a dialog will be displayed. This dialog will display the message provided during instantiation. In addition the following information of the client application will be displayed:

- the assembly title of the manifest data if available; otherwise, the process name
- the process ID
- the SID (Openness Session ID of the client application's TiaPortal instance)

Note

There can be multiple sessions active for a given TIA Portal Openness client application because there can be multiple instances of TiaPortal each associated with the same TIA Portal process.

The client application can also update the displayed content of the exclusive access dialog by setting the "Text" attribute with new values. Modify the following program code to invoke this behavior:

```
exclusiveAccess = ...;
...
exclusiveAccess.Text = "My Activity Phase 1";
...
exclusiveAccess.Text = "My Activity Phase 2";
...
exclusiveAccess.Text = String.Empty; // or null;
...
```

You can request that the exclusive access could be cancelled by selecting the "Cancel" button. Modify the following program code to invoke this behavior:

```
exclusiveAccess = ...;
...
if (exclusiveAccess.IsCancellationRequested)
{
    // stop your activity
    ...
}
else
{
    // continue your activity
    ...
}
...
...
```

7.9.9 Dynamic behaviours

Dynamic behaviors in Openness

The Openness users can determine objects and their attributes during the run time. They also are able to invoke actions and navigate to related objects bidirectionally in the hierarchy. In order to support dynamic behaviors the following interfaces are provided and implemented by Openness objects:

IEngineeringInstance

The interface that allows up navigation in the Object Model.

Properties:

- IEngineeringObject Parent - Parent object of an instance instance.

IEngineeringCompositionOrObject

The interface is implemented by both engineering objects and compositions.

IEngineeringObject

The interface is implemented by the majority of objects that are available for the Openness users. IEngineeringObject interface is derived from IEngineeringCompositionOrObject and IEngineeringInstance:

Methods:

- `IList<EngineeringCreateInfo> GetCreationInfos(string compositionName)` - Gets the list of composition infos available for the object.
- `IEngineeringObject Create(string compositionName, Type type, IEnumerable<KeyValuePair<string, object>> parameters)` - Creates an IEngineeringObject of indicated type initialized with values as indicated compositionName within the parameters
- `object GetAttribute(string name)` - Gets an attribute with the given name.
- `IList<EngineeringAttributeInfo> GetAttributeInfos()` - Returns a collection of EngineeringAttributeInfo objects describing the different attributes on this object.
- `IList<object> GetAttributes(IEnumerable<string> names)` - Gets a list of attributes for the given names
- `IEngineeringCompositionOrObject GetComposition(string name)` - Gets an IEngineeringCompositionOrObject with the given name.
- `IList<EngineeringInvocationInfo> GetInvocationInfos()` - Returns a collection of EngineeringInvocationInfo objects describing the different actions on this object.
- `object Invoke(string name, IEnumerable<KeyValuePair<Type, object>> parameters)` - Invokes the method represented by the current instance, using the specified parameters.
- `void SetAttribute(string name, object value)` - Sets an attribute with the given name to the given value
- `void SetAttributes(IEnumerable<KeyValuePair<string, object>> attributes)` - Sets the attributes with the given names to the given values as indicated in attributes.
- `EngineeringObjectHandle GetHandle()` - Gets the object's unique handle.

IEngineeringRoot

This interface is implemented by TiaPortal objects and indicates that the user is at the top object in the hierarchy. IEngineeringRoot is derived from IEngineeringObject, IEngineeringCompositionOrObject and IEngineeringInstance.

Methods:

- `IEngineeringObject GetObject(EngineeringObjectHandle objectHandle)` - Gets the object from a handle

IEngineeringComposition

The interface is implemented by the majority of objects that are available for the Openness users. IEngineeringObject interface is derived from IEngineeringCompositionOrObject and IEngineeringInstance. In addition to the properties and methods described in Working with compositions (Page 96), the following methods are supported.

Methods:

- IEngineeringObject Create(Type type, IEnumerable<KeyValuePair<string, object>> parameters) - Creates an IEngineeringObject of indicated type initialized with values as indicated in parameters..
- IList<EngineeringCreateInfo> GetCreationInfos() - Gets the collection of EngineeringCreateInfo objects describing the different CreateInfos on this object.
- IList<EngineeringInvocationInfo> GetInvocationInfos() - Returns a collection of EngineeringInvocationInfo objects describing the different actions on this object.
- object Invoke(string name, IEnumerable<KeyValuePair<Type, object>> parameters) - Invokes the method represented by the current instance, using the specified parameters.

IEngineeringObjectAssociation

This interface is implemented by the collection of associated engineering objects. This interface is derived from IEngineeringAssociation and IEngineeringInstance. See Working with association (Page 96)

IEngineeringServiceProvider

The interface is implemented by engineering objects that can provide service(s).

Methods:

- T GetService<T>() where T : class, IEngineeringService - Gets an instance of type T.
- IList<EngineeringServiceInfo> GetServiceInfos() - Returns a collection of EngineeringServiceInfo objects describing the different services on this object.

IEngineeringService

The interface representing an engineering service that is provided by IEngineeringServiceProvider.

EngineeringObjectHandle

The structure that provides a unique object handle.

See also

[Working with compositions \(Page 96\)](#)

[Working with associations \(Page 96\)](#)

7.9.10 Working with associations

Accessing associations

An association describes the relationship between two or more objects at type level.

TIA Portal Openness supports access to associations via index and via "foreach" loops. Direct access, for example via string name, is not supported.

Attributes

The following attributes are available:

- int Count
- bool Isreadonly
- IEngineeringObject Parent
- retType this [int index] { get; }

Methods

TIA Portal Openness supports the following methods:

- int IndexOf (type): Returns the index in the association for a transferred instance.
- bool Contains (type): Determines whether the transferred instance is contained in the association.
- IEnumator GetEnumator <retType>(): Employed within "foreach" loops to access an object.
- void Add (type)¹: Adds the transferred instance to the association.
- void Remove (type)¹: Removes the transferred instance from the association.

¹: Not supported by all associations.

7.9.11 Working with compositions

Accessing compositions

A composition is the special case of an association. A composition expresses a semantic relationship of two objects, of which one is part of the other.

Attributes

The following attributes are available:

- int Count
- bool Isreadonly

- `IEngineeringObject Parent`
- `retType this [int index] {get; };`: Indexed access to an object of the composition. This type of access should only be used in a targeted manner, as each indexed access operation exceeds process boundaries.

Methods

TIA Portal Openness supports the following methods:

- `retType Create (id, ...);`: Creates a new instance and adds this instance to the composition.
The signature of the method depends on the way in which the instance is created. This method is not supported by all compositions.
- `type Find (id, ...);`: Scans a composition for the instance with the transferred ID. The search is not recursive. The signature of the method depends on the way in which the instance is searched for. This method is not supported by all compositions.
- `IEnumerator<retType> GetEnumerator ()`: Employed within "foreach" loops to access an object.
- `Delete (type)`¹: Deletes the instance specified by the current object reference.
- `int IndexOf (type)`: Returns the index in the composition for a transferred instance.
- `bool Contains (type)`: Determines whether the transferred instance is contained in the composition.
- `void Import(string path, ImportOptions importOptions)`¹: Used for each composition that contains importable types.
Each import signature includes a configuration parameter of the type "ImportOptions (Page 753)" ("None", "Overwrite") by which the user controls the import behavior.

¹: Not supported by all compositions.

7.9.12 Verifying object equality

Application

As user of a TIA Portal Openness API, you can check that objects are the same with program code:

- You check whether two object references are the same with the operator "`==`".
- Use the `System.Object.Equals()` method to check if both objects are really identical with regard to the TIA Portal.

Program code

Modify the following program code to check for object reference types:

```
...
//Composition
DeviceComposition sameCompA = project.Devices;
DeviceComposition sameCompB = project.Devices;
if (sameCompA.Equals(sameCompB))
{
    Console.WriteLine("sameCompA is equal to sameCompB");
}
if (!(sameCompA == sameCompB))
{
    Console.WriteLine("sameCompA is not reference equal to sameCompB");
}
DeviceComposition sameCompAsA = sameCompA;
if (sameCompAsA.Equals(sameCompA))
{
    Console.WriteLine("sameCompAsA is equal to sameCompA");
}
if (sameCompAsA == sameCompA)
{
    Console.WriteLine("sameCompAsA is reference equal to sameCompA");
}
MultiLingualGraphicComposition notSameComp = project.Graphics;
if (!sameCompA.Equals(notSameComp))
{
    Console.WriteLine("sameCompA is not equal to notSameComp");
}
```

7.9.13 Read operations for attributes

Group operations and standard read operations for attributes

TIA Portal Openness supports access to attributes via the following methods which are available at the object level:

- Group operation for read access
- Standard read operations

Program code for group operations

```
//Exercise GetAttributes and GetAttributeNames
//get all available attributes for a device,
//then get the names for those attributes, then display the results.
private static void DynamicTest(Project project)
{
    Device device = project.Devices[0];
    IList<string> attributeNames = new List<string>();
    IList<EngineeringAttributeInfo> attributes =
((IEngineeringObject)device).GetAttributeInfos();
    foreach (EngineeringAttributeInfo engineeringAttributeInfo in attributes)
    {
        string name = engineeringAttributeInfo.Name;
        attributeNames.Add(name);
    }
    IList<object> values = ((IEngineeringObject)device).GetAttributes(attributeNames);
    for (int i = 0; i < attributes.Count; i++)
    {
        Console.WriteLine("attribute name: " + attributeNames[i] + " value: " + values[i]);
    }
}
```

Group operation for read access

This method is available for any object:

```
public abstract IList<object> GetAttributes(IEnumerable<string>
names);
```

Standard read operations

The following operations are available:

- Retrieve the names of available attributes:
Use the method `GetAttributeInfos()` (Page 103) on an `IEngineeringObject`.
- Generic method for reading an attribute
`public abstract object GetAttribute(string name);`

Note

Dynamic attributes are not shown in IntelliSense because their availability depends on the status of the object instance.

7.9.14 Transaction handling

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Operation

A persistence (project, library, etc.) opened within an associated TIA Portal process can be modified by a TIA Portal Openness client application. You can produce this modification from a single operation or by a series of operations. Depending on the activity, it is reasonable to group these operations into a single undo unit for more logical workflows. Additionally there are performance advantages provided by grouping operations into a single undo unit. To support this, the "ExclusiveAccess" class provides the method "Transaction(ITransactionSupport persistence, string undoDescription)". The invocation of this method results in the instantiation of a new disposable object of type "Transaction". You have to provide a description of the transaction's contents (the text attribute cannot be null or Empty). While this instance has not been disposed, all client application operations will be grouped into a in a single undo unit within the associated TIA Portal process.

Modify the following program code to acquire a "Transaction" instance:

```
ExclusiveAccess exclusiveAccess = ...;
Project project = ...;
using (Transaction transaction = exclusiveAccess.Transaction(project, "My Operation"))
{
    ...
}
```

Note

Use a "using" statement to instantiate a "Transaction" to ensure it is disposed properly even when exceptions occur, thus rolling back the transaction.

Consistent commit or rollback

The use of a "Transaction" within a client application helps you to ensure that there is a predictable way to commit or rollback a set of modifications. Your client application must decide whether or not to commit their modifications to a persistence. To do this your application must request that the modifications within the scope of an open transaction be committed when the transaction is disposed by invoking the 'Transaction.CommitOnDispose()' method. If this method is never invoked in the code flow, the modifications within the scope of the open transaction will automatically be rolled back when it is disposed.

If an exception occurs after making the request, all modifications within the scope of an open transaction will still be rolled back on its disposal.

Modify the following program code to create a single undo unit in the attached TIA Portal containing two "Create" modifications:

```
ExclusiveAccess exclusiveAccess = ...;
Project project = ...;
using (Transaction transaction = exclusiveAccess.Transaction(project, "My Operation"))
{
    project.DeviceGroups.Create("My Group 1");
    project.DeviceGroups.Create("My Group 2");
    transaction.CommitOnDispose();
}
```

Restrictions

The following actions are not allowed inside of a transaction. Calling these will result in a recoverable exception:

- Compile
- Go Online
- Go Offline
- ProjectText Import
- ProjectText Export
- Open Global Library
- Close Global Library
- Project Create
- Project Open
- Project OpenWithUpgrade
- Project Save
- Project SaveAs
- Project Close

Undo behavior

Actions performed by a TIA Portal Openness client application can result in undo units within the attached TIA Portal process. Each of these undo entries will be grouped under a location entry. This location entry will compose the following information from the client application:

- the assembly title from the manifest data if available; otherwise, the process name
- the process ID
- the SID (Openness Session ID of the client application's TiaPortal instance)
- optionally an indication that the client process is still running

These entries will be one of the following two kinds:

1. The operations that are gathered into one undo transaction as a result of using a "Transaction" have the description as provided by the client application when the "Transaction" was instantiated.

- Undo entry for a running client application:

 MySuperApplication ,PID: 4704, SID: 4, running
 My Operation

- Undo entry for a stopped client application:

 MySuperApplication, PID: 4704, SID: 4
 My Operation

2. The operations that are executed individually have individual undo entries describing the operation as defined in the respective command meta data.

- Undo entry for a running client application:

 MySuperApplication ,PID: 11224, SID: 3, running
 Create Folder "My Folder 2"
 Create Folder "My Folder 1"

- Undo entry for a stopped client application:

 MySuperApplication, PID: 11224, SID: 3
 Create Folder "My Folder 2"
 Create Folder "My Folder 1"

7.9.15 Creating a DirectoryInfo/FileInfo object

Application

The instances of `DirectoryInfo` and `FileInfo` classes have to contain an absolute path. Otherwise the methods using the `DirectoryInfo` or `FileInfo` objects will lead to an exception.

Program code

Modify the following program code to create a `DirectoryInfo` or a `FileInfo` object.

```
...
//Create a DirectoryInfo object
string directoryPath = @"D:\Test\Project 1";
DirectoryInfo DirectoryInfo = new DirectoryInfo(directoryPath);

//Create a FileInfo object
//please adapt the path and the extension apx to the installed version of TIA Portal
string fileName = @"D:\Test\Project 1\Project 1.apx";
FileInfo fileInfo = new FileInfo(fileName);
...
...
```

7.9.16 Self-description support for attributes, navigators, actions, and services

Application

In TIA Portal Openness each `IEngineeringServiceProvider` of the TIA Portal Openness API describes its capabilities to potential calls.

Self-description Support on `IEngineeringObject`

Method Name	Return values
<code>GetCompositionInfos</code>	Returns a collection of <code>EngineeringCompositionInfo</code> objects describing the different compositions of these objects. <code>EngineeringCompositionInfo</code> is described below.
<code>GetAttributeInfos</code>	Returns a collection of <code>EngineeringAttributeInfo</code> objects describing the different attributes of these objects. <code>EngineeringAttributeInfo</code> is described below.
<code>GetInvocationInfos</code>	Returns a collection of <code>EngineeringInvocationInfo</code> objects describing the different actions of these objects. <code>EngineeringInvocationInfo</code> is described below.

Self-description Support on `IEngineeringServiceProvider`

Method Name	Return values
<code>GetServiceInfos</code>	Returns a collection of <code>EngineeringServiceInfo</code> objects describing the different services of these objects. <code>EngineeringServiceInfo</code> is described below.

Self-description Support on IEngineeringComposition

Method Name	Return values
GetCreationInfos	Returns a collection of EngineeringCreationInfo objects describing the different actions of these objects. EngineeringcreationInfo is described below.

Class EngineeringCompositionInfo

Attribute Name	Return values
Name	The name of the composition

Class EngineeringAttributeInfo

Attribute Name	Return values
AccessMode	The level of access supported by the attribute. This attribute is combinable and is described in detail below.
Name	The name of the attribute.

Class EngineeringInvocationInfo

Attribute Name	Return values
Name	The name of the action.
ParameterInfos	A collection of EngineeringInvocationParameterInfo objects describing any parameters that the action might require. EngineeringInvocationParameterInfo is described below.

Class EngineeringServiceInfo

Attribute Name	Return values
Type	The type of the service as a System.Type object.

Enum AccessMode

Enum Value	Return values
None	This is not a valid option.
Read	The attribute can be read.
Write	The attribute can be written.

Class EngineeringInvocationParameterInfo

Attribute Name	Return values
Name	The name of the parameter.
Type	The type of the parameter as a System.Type object

Class EngineeringCreationInfo

Attribute Name	Return values
Type	The Type property returns a System.Type object
ParameterInfos	The ParameterInfos property returns a collection of EngineeringCreationParameterInfo objects

Class EngineeringCreationParameterInfo

Attribute Name	Return values
Name	The name of the parameter in question

Program code

AccessMode is a flags enum and its values can be combined like the following program code:

```
EngineeringAttributeAccessMode value = EngineeringAttributeAccessMode.Read |  
EngineeringAttributeAccessMode.Write;
```

Modify the following program code to find all attributes of an IEngineeringObject and to do changes on the access mode of those attributes.

```
...  
IEngineeringObject engineeringObject = ...;  
IList<EngineeringAttributeInfo> attributeInfos = engineeringObject.GetAttributeInfos();  
foreach(EngineeringAttributeInfo attributeInfo in attributeInfos)  
{  
    switch (attributeInfo.AccessMode)  
    {  
        case EngineeringAttributeAccessMode.Read:  
            ...  
            break;  
        case EngineeringAttributeAccessMode.Write:  
            ...  
            break;  
        case EngineeringAttributeAccessMode.Read | EngineeringAttributeAccessMode.Write:  
            ...  
            break;  
    }  
}
```

7.10 Functions for projects and project data

Modify the following program code of how Create and GetCreationInfos are intended to be used together.

```
...
Project project = tiaPortal.Projects.Open(projectPath);
IEngineeringComposition deviceGroupComposition = project.DeviceGroups;
EngineeringCreateInfo deviceGroupCreateInfo = deviceGroupComposition.GetCreationInfos()[0];
EngineeringCreationParameterInfo deviceGroupCreationParameterInfo =
deviceGroupCreateInfo.ParameterInfos[0];
string deviceGroupCreationParameterName = deviceGroupCreationParameterInfo.Name;
string groupName = "Example";
IDictionary<string, object> createParameters = new Dictionary<string, object>
{
    {deviceGroupCreationParameterName, groupName }
};
IEngineeringObject group = deviceGroupComposition.Create(deviceGroupCreateInfo.Type,
createParameters);
...
...
```

7.10 Functions for projects and project data

7.10.1 Opening a project

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- The project to be opened is not open in any other instance of the TIA Portal.

Note

Undo of project upgrade

If you undo an upgrade of a project to V14SP1 after you have connected it to TIA Portal Openness, conflicts will occur.

Application

Use the `Projects.Open` method to open a project. Enter a path to the desired project in the `Projects.Open` method.

The `Projects.Open` method only accesses projects that were created with the current version of TIA Portal or which have been upgraded to the current version. If you access a project of a previous version with the `Projects.Open` method, an exception will be returned.

Use the `OpenWithUpgrade` method, to open projects which have been made with previous versions of TIA Portal.

Note**No access to read-only projects**

TIA Portal Openness can only access projects with read and write privileges.

Program code

Modify the following program code to open a project:

```
Project project =tiaPortal.Projects.Open(new FileInfo(@"D:\Project_1\Project_1.apXX")) ;
if (project != null)
{
    try
    {
        ...
    }
    finally
    {
        project.Close();
    }
}
```

Opening a UMAC protected project

You can also open a UMAC protected project. The overload of Open function takes an additional parameter of type `UmacDelegate`. This additional parameter allows the caller to specify a handler to be used during UMAC authentication. The new `UmacDelegate` is implemented with a method containing one parameter of type `UmacCredentials`. The `UmacCredentials` has two properties, 'Name' of type string, and 'Type' of type `UmacUserType`; and one method `SetPassword` with one parameter of type `SecureString`. The use of `UmacUserType.Project` indicates a UMAC scope of project whereas the use of `UmacUserType.Global` indicates a UMAC scope of application (i.e. controlled by a UMAC server).

Program code

```
...
    Siemens.Engineering.Project project = tiaPortal.Projects.Open(new
FileInfo(@"D:\Project_3\Project_2.apXX"), MyUmacDelegate);
    if (project != null)
    {
        try
        {
            ...
        }
    finally
    {
        project.Close();
    }
}
...
private static void MyUmacDelegate(UmacCredentials umacCredentials)
{
    SecureString password = ...; // Get password from a secure location
    umacCredentials.Type = UmacUserType.Project;
    umacCredentials.Name = "SomeUser";
    umacCredentials.SetPassword(password);
}
...
}
```

Opening multiple projects

You can open one primary project and multiple secondary projects at time in an instance of the TIA Portal. In this case you have to decide to open a project as a primary or a secondary project. If a project is opened as primary, then this project is represented in the project navigation if the Openness application is attached to a TIA Portal. If a project is opened as secondary, the project will not be reflected in the user interface. Secondary projects are always opened as read-only. Therefore, a user with read and write privileges to UMAC projected project will only have read-only privileges when project is opened as secondary. A primary project does not need to be open in order to open a secondary project.

Any opened projects can be enumerated by using the ProjectComposition available on the TiaPortal instance. The order of the projects in the composition will be determined by the order in which the projects were opened. If a project is closed, the index of all projects will be recalculated.

Program code

```
TiaPortal tiaPortal = ...;
Project project1 = tiaPortal.Projects.Open(new
FileInfo(@"D:\Project_1\Project_1.apXX"), null, ProjectOpenMode.Primary);
Project project3 = tiaPortal.Projects.Open(new
FileInfo(@"D:\Project_3\Project_3.apXX"), null, ProjectOpenMode.Secondary);
bool isPrimary = project3.IsPrimary
```

Opening projects created with previous versions

Use the `OpenWithUpgrade` method to open a project which was created with the previous version of TIA Portal. The method will create a new, upgraded project and open it.

If you access a project created with an elder version than the previous, an exception will be returned.

Note

If you access a project created with the current version the project will just be opened.

Program code

Modify the following program code to open a project via the `OpenWithUpgrade` method:

```
Project project = tiaPortal.Projects.OpenWithUpgrade(new FileInfo(@"D:\Some
\Path\Here\Project.apXX"));
if (project != null)
{
    try
    {
        ...
    }
    finally
    {
        project.Close();
    }
}
```

Program code for UMAC protected project

You can also open a UMAC protected which has been created with a previous version of TIA Portal. An overload function of `OpenWithUpgrade` takes an additional parameter of type `UmacDelegate`. `OpenWithUpgrade` is also supported for secondary projects.

```
...
Siemens.Engineering.Project project = tiaPortal.Projects.OpenWithUpgrade(new
FileInfo(@"D:\Project_1\Project.apXX"), MyUmacDelegate);
...
```

7.10.2 Creating a project

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)

Application

Projects can be created via TIA Portal Openness API

- by calling the Create method on ProjectComposition
- by calling the Create method on IEngineeringComposition

ProjectComposition.Create

Modify the following program code:

```
TiaPortal tiaPortal = ....;
ProjectComposition projectComposition = tiaPortal.Projects;
DirectoryInfo targetDirectory = new DirectoryInfo(@"D:\TiaProjects");

// Create a project with name MyProject
Project project = projectComposition.Create(targetDirectory, "MyProject");
```

According to this example

- a folder "D:\TiaProjects\MyProject" will be created.
- a project file "D:\TiaProjects\MyProject\MyProject.aPXX" will be created.

Note

About parameter targetDirectory

The parameter targetDirectory can also represent an UNC (Universal Naming Convention) path, therefore a project can also be created on a network shared drive.

IEngineeringComposition.Create

Modify the following program code:

```

TiaPortal tiaPortal = ...;
ProjectComposition projectComposition = tiaPortal.Projects;

//allows the user to give optional create parameters like author, comment in addition to
mandatory create parameters (targetdirectory, projectname)

IEnumerable<KeyValuePair<string, object>> createParameters = new [] {
    new KeyValuePair<string, object>("TargetDirectory", new
DirectoryInfo(@"D:\TiaProjects")), // Mandatory
    new KeyValuePair<string, object>("Name", "MyProject"), // Mandatory
    new KeyValuePair<string, object>("Author", "Bob"), // Optional
    new KeyValuePair<string, object>("Comment", "This project was created with
Openness") // Optional };

// Create a project with both mandatory and optional parameters
((IEngineeringComposition)projectComposition).Create(typeof (Project), createParameters);

```

According to this example

- a folder "D:\TiaProjects\MyProject" will be created.
- a project file "D:\TiaProjects\MyProject\MyProject.aPXX" will be created with project attributes Author as "Bob" and Comment as "This project was created with openness".

Parameters for creating project with optional project attributes

Parameter	Data Type	Is Mandatory	Description
Author	String	No	Author of a project.
Comment	String	No	Comment for the project.
Name	String	Yes	Name of a project,
TargetDirectory	DirectoryInfo	Yes	Directory that will contain the created project folder.

7.10.3 Accessing general settings of the TIA Portal

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

Via TIA Portal Openness you can access general settings of the TIA Portal:

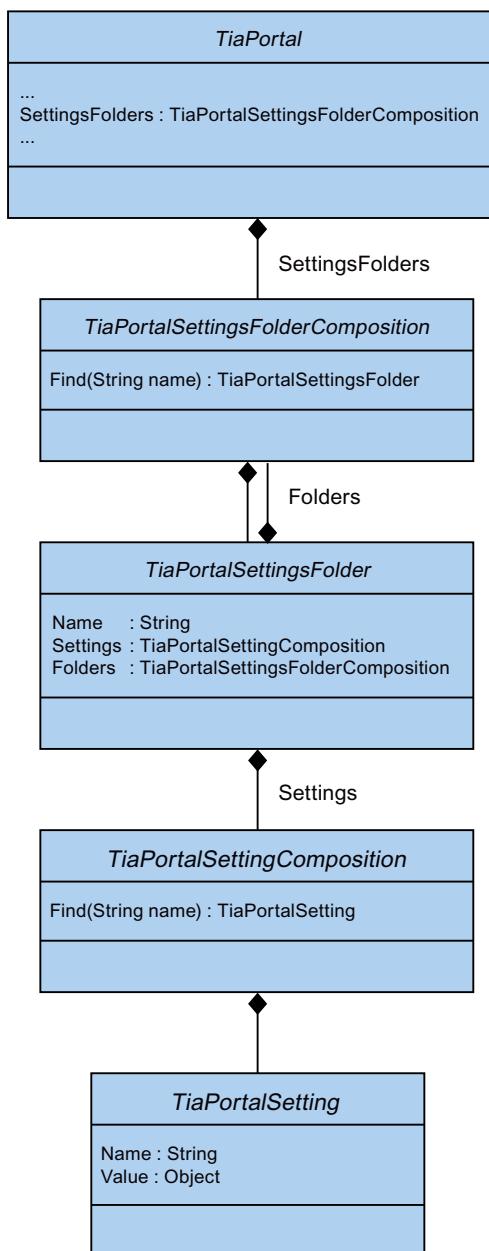
- Current user interface language
- "Search in project" option to create the search index needed for searching within a project.

The following table shows the details of the accessible settings in the "General" section of the TIA Portal settings. The `TiaPortalSettingsFolder` instance will have the name "General".

Setting name	Data type	Writable	Description
"SearchInProject"	System.Boolean	r/w	Enables or disables the creation of the search index needed for searching within a project.
"UserInterfaceLanguage"	System.CultureInfo	r/w	Indicates the active user interface language of the TIA Portal or the specification of the active user interface language.

The access to these settings is provided via the `TiaPortalSettingsFolder` class. The `TiaPortalSettingsFolder` class will be accessible via the `Settings` attribute on the `TiaPortal` class.

The following figure shows the specific settings in TIA Portal Openness:



Program code: Search in project

Modify the following program code to activate/deactivate the "Search in project" option.

```
private static void SetSearchInProject(Project project)
{
    TiaPortalSettingsFolder generalSettingsFolder =
tiaPortal.SettingsFolders.Find("General");
    TiaPortalSetting searchSetting =
generalSettingsFolder.Settings.Find("SearchInProject");

    if (((bool)searchSetting.Value))
    {
        searchSetting.Value = false;
    }
}
```

Program code: User interface language

Modify the following program code to access the current user interface language.

```
private static void SetUILanguage(Project project)
{
    TiaPortalSettingsFolder generalSettingsFolder =
tiaPortal.SettingsFolders.Find("General");

    TiaPortalSetting UILanguageSetting =
generalSettingsFolder.Settings.Find("UserInterfaceLanguage");

    if (((CultureInfo)UILanguageSetting.Value) != CultureInfo.GetCultureInfo("de-DE"))
    {
        UILanguageSetting .Value = CultureInfo.GetCultureInfo("de-DE");
    }
}
```

See also

[Opening a project \(Page 106\)](#)

7.10.4 Archiving and Retrieving a project

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a Project (Page 106)
- A project is save
See Saving a Project (Page 131)

Application

You can use the TIA Portal Openness to archive the opened and saved project state prior to further any changes and then later you can retrieve the archived project.

Archiving a project

You can archive a TIA Portal project using TIA Openness API available on Siemens.Engineering.Project object.

```
public void Archive(System.IO.DirectoryInfo targetDirectory, string targetName,  
Siemens.Engineering.ProjectArchivationMode archivationMode)
```

'targetName' is the name of the file created may it be archived or non archived. This file may or may not contain any file extensions. For ProjectArchivationMode value as Compressed and DiscardRestorableDataAndCompressed, the archived file name is as it is provided by you. If you don't provide any extension or extension apart from "zap15_1" or "zap14" etc., then this archived project could not be retrieved from TIA Portal outside the Openness API. If you don't not choose Compressed or DiscardRestorableDataAndCompressed the result will be the default file structure of a project of the current version.

Note

You must have saved the project before calling Archiving API. In case the project contains any unsaved changes, archive would throw an EngineeringTargetInvocationException.

Project ArchivationMode

The ProjectArchivationMode enumeration have four values.

ProjectArchivationMode	Description
None	<ul style="list-style-type: none"> No special action are taken with the orginal files. Mode is similiar to a "save as" operation. No compressed zip file is created in this mode. The difference with SaveAs in this case is that the Archive will not change the persistence location to the new Archived folder whereas SaveAs does that
DiscardRestorableData	<ul style="list-style-type: none"> The file storage stores the project in an internal data file and this file grows whenever a project data modification happens. In the case of DiscardRestorableData mode this data file is reorganized (only latest version of the objects are stored and history is removed from the file) and Intermediate data, the files of the IM directory and tmp directory are not copied to the archive location. No compressed zip file is created in this mode.
Compressed	<ul style="list-style-type: none"> The tmp project folder structure, created by Archiving, is compressed into a zip compatible archive. The tmp folder structure is removed after creation of the zip file.
DiscardRestorableDataAndCompressed	<ul style="list-style-type: none"> In the tmp project folder structure, created by Archiving, the restorable data is discarded and then compressed into a zip compatible archive. The tmp folder structure is removed after creation of the zip file.

Program code: Archiving a project

Modify the following program code to access the archive project without an extension:

```
var tiaPortal = new TiaPortal(TiaPortalMode.WithoutUserInterface);
var projectFilePath = @"E:\Sample1\Sample1.ap15_1";
var project = tiaPortal.Projects.Open(new FileInfo(projectFilePath));
var archiveDirectory = @"E:\Archive";
project.Archive(new DirectoryInfo(archiveDirectory), "ArchiveProjectNameWithoutExtension",
ProjectArchivationMode.Compressed);
```

Modify the following program code to access the archive project with an extension as '.archive':

```
var tiaPortal = new TiaPortal(TiaPortalMode.WithoutUserInterface);
var projectFilePath = @"E:\Sample1\Sample1.ap15_1";
var project = tiaPortal.Projects.Open(new FileInfo(projectFilePath));
var archiveDirectory = @"E:\Archive";
project.Archive(new DirectoryInfo(archiveDirectory), "ArchiveProjectName.archive",
ProjectArchivationMode.Compressed);
```

Note

You are free to use any extension. The behavior of Archive would be same with/without extension.

Retrieving a project

You can use the Openness API to retrieve an archived TIA Portal project. You can only retrieve a compressed archive. For Archived project with 'ProjectArchivationMode.None' or 'ProjectArchivationMode.DiscardRestorableData' enumeration value cannot be retrieved by this API. This API is available on the "Siemens.Engineering.ProjectComposition" object.

```
public Siemens.Engineering.Project Retrieve(System.IO.FileInfo sourcePath,
System.IO.DirectoryInfo targetDirectory)
```

For retrieving a UMAC protected project an overloaded API with the UmacDelegate is provided for getting the credentials, then the following API definition is used

```
public Siemens.Engineering.Project Retrieve(System.IO.FileInfo sourcePath,
System.IO.DirectoryInfo targetDirectory, Siemens.Engineering.UmacDelegate umacDelegate)
```

Retrieve also supports ProjectOpenMode that can be specified as "Primary" or "Secondary", then the following API definition is used

```
public Siemens.Engineering.Project Retrieve(System.IO.FileInfo sourcePath,
System.IO.DirectoryInfo targetDirectory, Siemens.Engineering.UmacDelegate umacDelegate,
Siemens.Engineering.ProjectOpenMode projectOpenMode)
```

Note

umacDelegate could be passed as null if the project is not protected

If the archived project is of a previous TIA Portal version then you have to call the RetrieveWithUpgrade API, then the following API definition is used

```
public Siemens.Engineering.Project RetrieveWithUpgrade(System.IO.FileInfo sourcePath,
System.IO.DirectoryInfo targetDirectory);
```

If the archived project is of a previous TIA Portal version and UMAC protected then another overloaded API with UmacDelegate is available, then the following API definition is used.

```
public Siemens.Engineering.Project RetrieveWithUpgrade(System.IO.FileInfo sourcePath,
System.IO.DirectoryInfo targetDirectory, Siemens.Engineering.UmacDelegate umacDelegate)
```

7.10 Functions for projects and project data

If the archived project is of a previous TIA portal version and project open mode also needs to be specified, then the following API definition is used.

```
public Siemens.Engineering.Project RetrieveWithUpgrade(System.IO.FileInfo sourcePath,  
System.IO.DirectoryInfo targetDirectory, Siemens.Engineering.UmacDelegate umacDelegate,  
Siemens.Engineering.ProjectOpenMode projectOpenMode)
```

Note

umacDelegate could be passed as null if the project is not protected. If the project is protected then the user credentials of an admin user is required to perform the RetrieveWithUpgrade action

Program code: Retrieving a project

Modify the following program code to access the retrieve project:

```
var archivePath = @"E:\Archive\Sample1.zap15_1";  
var retrievedProjectsDirectory = @"E:\RetrievedProjects";  
var tiaPortal = new TiaPortal(TiaPortalMode.WithoutUserInterface);  
tiaPortal.Projects.Retrieve(new FileInfo(archivePath), new  
DirectoryInfo(retrievedProjectsDirectory));
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

[Saving a project \(Page 131\)](#)

7.10.5 Accessing read-only TIA Portal project

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open
See [Opening a project \(Page 106\)](#)

Application

Using TIA Portal Openness, you can perform select operations while working with a read-only TIA Portal project. You can have access to read-only project, but you will not be able to use full set of features that are available to a user with read-write access. For example, a user with read-only credentials can use Openness to open a UMAC protected project as described in Opening a project (Page 106). This functionality does not include Reference projects.

The list of Openness features that are available to you while accessing a read-only project can be categorized into two sets of features - Inherent and Enabled non-modifying actions:

Inherent functionality

- GetAttribute(s) or using the getter for any attribute on any accessible object
- GetComposition on any accessible object
- GetService on any accessible object
- Find actions on any accessible object
- Navigation on any accessible object
- Determining the existence of accessible objects and accessing those objects in compositions and associations
- System.Object methods on any accessible object

Enabled non-modifying actions

- Project.Close (...)
- PlcBlock.ShowInEditor ()
- CaxProvider.Export (Device,...)
- CaxProvider.Export (Project,...)

See also

[Opening a project \(Page 106\)](#)

7.10.6 Accessing languages

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)

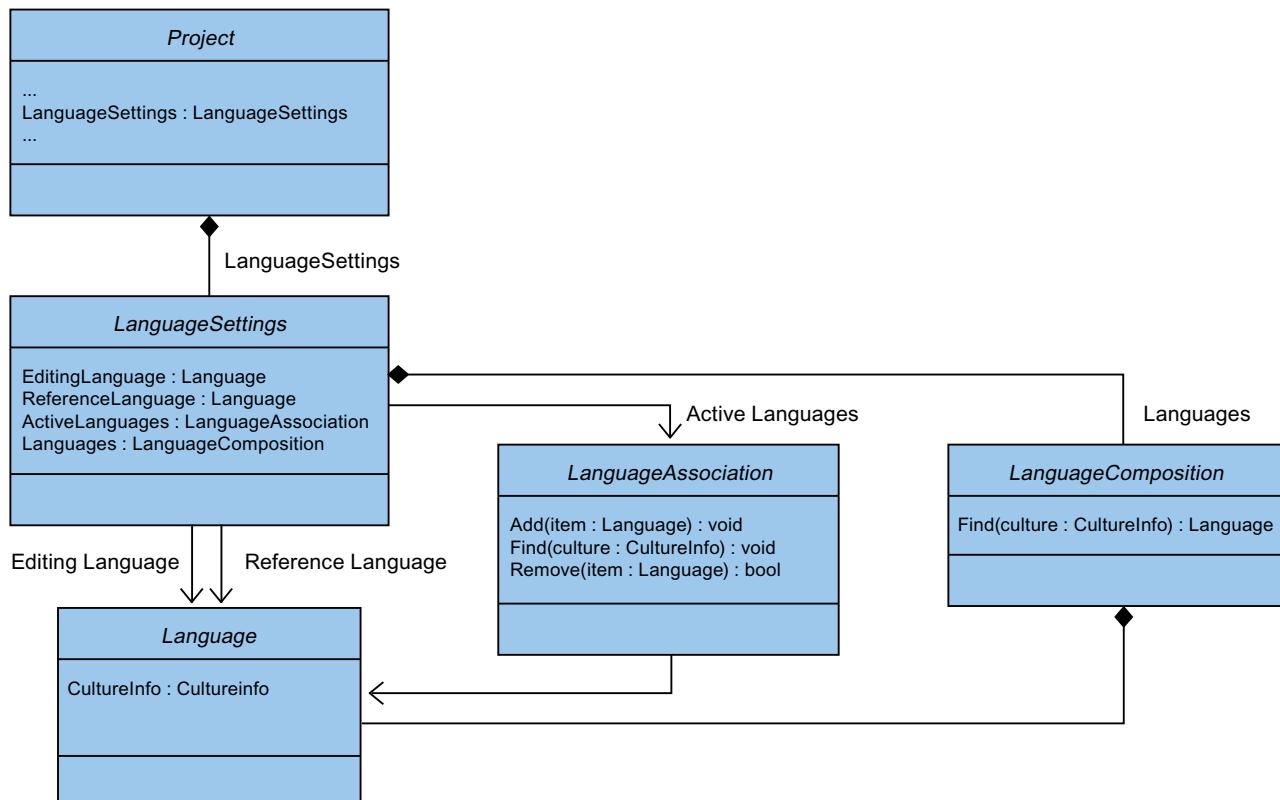
Application

In the TIA Portal you can set and manage the project language in the Editor "Project languages".

TIA Portal Openness supports the following access to the project languages:

- Iterating through supported languages.
- Searching through the collection of supported languages by `System.Globalization.CultureInfo`.
- Accessing individual languages. Each `Language` object will contain a single read-only attribute `Culture` of type `System.Globalization.CultureInfo`.
- Accessing a collection of active languages.
- Searchring through the collection of active languages by `System.Globalization.CultureInfo`.
- Adding a language to a collection of active languages.
- Removing a language from a collection of active languages.
- Setting an editing language.
- Setting a reference language.

The functionalities are provided by the `LanguageSettings` object. The following figure shows model, which is provided by TIA Portal Openness:



Program code: Setting languages

Modify the following program code to set a language. If you set an inactive language via TIA Portal Openness, the language will be added to the active languages collection.

```
Project project = ...;

LanguageSettings languageSettings = project.LanguageSettings;

LanguageComposition supportedLanguages = languageSettings.Languages;
LanguageAssociation activeLanguages = languageSettings.ActiveLanguages;

Language supportedGermanLanguage =
supportedLanguages.Find(CultureInfo.GetCultureInfo("de-DE"));
activeLanguages.Add(supportedGermanLanguage);

languageSettings.EditingLanguage = supportedGermanLanguage;
languageSettings.ReferenceLanguage = supportedGermanLanguage;
```

Program code: Deactivating an active language

Modify the following program code to deactivate an active language. If you deactivate a language which is used as reference or editing language the language selected will be consistent with the behavior in the user interface.

```
Project project = ...;

LanguageSettings languageSettings = project.LanguageSettings;
LanguageAssociation activeLanguages = languageSettings.ActiveLanguages;
Language activeGermanLanguage = activeLanguages.Find(CultureInfo.GetCultureInfo("de-DE"));
activeLanguages.Remove(activeGermanLanguage);
```

7.10.7 Determining the object structure and attributes

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- You have opened a project with your TIA Portal Openness application.
See Opening a project (Page 106)

Application

You can determine the navigation structure through the object hierarchy with the IEngineeringObject interface. The result is returned as a list:

- Child objects
- Child compositions
- All attributes

Signature

Use the GetAttributeInfos method to determine attributes.

```
IList<EngineeringAttributeInfo>
IEngineeringObject.GetAttributeInfos();
```

Program code: Determining objects or compositions

Use the following program code to display all composition names:

```
public static void DisplayCompositionInfos(IEngineeringObject obj)
{
    IList<EngineeringCompositionInfo> compositionInfos = obj.GetCompositionInfos();
    foreach (EngineeringCompositionInfo compositionInfo in compositionInfos)
    {
        Console.WriteLine(compositionInfo.Name);
    }
}
```

Modify the following program code if you know the return value:

```
public static DeviceItemComposition GetDeviceItemComposition(Device device)
{
    IEngineeringCompositionOrObject composition = ((IEngineeringObject)
device).GetComposition("DeviceItems");
    DeviceItemComposition deviceItemComposition = (DeviceItemComposition)composition;
    return deviceItemComposition;
}
```

Program code: Determining attributes

Modify the following program code to return attributes of an object with specific access rights in a list:

```
public static void DisplayAttributeInfos(IEngineeringObject obj)
{
    IList<EngineeringAttributeInfo> attributeInfos = obj.GetAttributeInfos();
    foreach (EngineeringAttributeInfo attributeInfo in attributeInfos)
    {
        Console.WriteLine("Attribute: {0} - AccessMode {1} ",
            attributeInfo.Name, attributeInfo.AccessMode);
        switch (attributeInfo.AccessMode)
        {
            case EngineeringAttributeAccessMode.Read: Console.WriteLine("Attribute: {0} - Read Access", attributeInfo.Name);
                break;
            case EngineeringAttributeAccessMode.Write: Console.WriteLine("Attribute: {0} - Write Access", attributeInfo.Name);
                break;
            case EngineeringAttributeAccessMode.Read | EngineeringAttributeAccessMode.Write:
                Console.WriteLine("Attribute: {0} - Read and Write Access", attributeInfo.Name);
                break;
        }
    }
}

public static string GetNameAttribute(IEngineeringObject obj)
{
    Object nameAttribute = obj.GetAttribute("Name");
    return (string)nameAttribute;
}
```

7.10.8 Access software target

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Programm code

Modify the following programm code to make a software target available:

```
SoftwareContainer softwareContainer =
((IEngineeringServiceProvider)deviceItem).GetService<SoftwareContainer>();
if (softwareContainer != null)
{
    Software software = softwareContainer.Software;
}
```

Modify the following programm code to access the software attributes:

```
SoftwareContainer softwareContainer =
((IEngineeringServiceProvider)deviceItem).GetService<SoftwareContainer>();
if (softwareContainer != null)
{
    PlcSoftware software = softwareContainer.Software as PlcSoftware;
    string name = software.Name;
}
```

7.10.9 Accessing and enumerating multilingual texts

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

Multilingual texts in the TIA Portal are e. g. Project.Comment, PlcTag.Comment etc. In TIA Portal Openness, the multilingual texts are represented by the `MultilingualText` object. A `MultilingualText` object is composed of `MultilingualTextItemComposition`.

`MultilingualTextItemComposition` supports the following `Find` method:

- `Find(<language:
Siemens.Engineering.Language>):MultilingualTextItem`

Each `MultilingualTextItem` provides the following attributes:

Attribute name	Data type	Writable	Description
Language	Siemens.Engineering.Language	r/o	Language of this item.
Text	System.String	r/w	Text provided for this language.

Program code: Set multilingual text

```
...
    Language englishLanguage = project.LanguageSettings.Languages.Find(new CultureInfo("en-US"));
    MultilingualText comment = project.Comment;
    MultilingualTextItemComposition mltItemComposition = comment.Items;
    MultilingualTextItem englishComment = mltItemComposition.Find(englishLanguage);
    englishComment.Text = "English comment";
...

```

Program code: Set multilingual text for devices

Modify the following program code to set the multilingual text for devices and device items:

```
...
var mltObject = device.GetAttribute("CommentML");
MultilingualText multilingualText = mltObject as MultilingualText;
if (multilingualText != null)
{
    Language englishLanguage = project.LanguageSettings.Languages.Find(new CultureInfo("en-US"));
    MultilingualTextItem multilingualTextItem =
multilingualText.Items.Find(englishLanguage);
    if (multilingualTextItem != null)
    {
        multilingualTextItem.Text = comment;
    }
}
...

```

7.10.10 Updating project property simulation support

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)

Application

You can use the TIA Portal Openness to update the project property "simulation support" so that you can create plc programs that are also used in a virtual SINUMERIK controller.

7.10 Functions for projects and project data

The following attributes are supported by Openness to read/write the property "simulation support" of a TIA Portal project. The API is available on the "Siemens.Engineering.Project" object.

Attribute name	Data type	Write	Description
IsSimulationDuringBlockCompilationEnabled	System.Boolean	yes	To indicate whether Support for Simulation during block compilation is enabled for the project

Program code

Modify and use the following program code to read/write the attribute value:

```
Project project = ...;
// Read the attribute value
var attributeValue = project.IsSimulationDuringBlockCompilationEnabled;
//Write the attribute value to false
project.IsSimulationDuringBlockCompilationEnabled = false;
```

See also

[Opening a project \(Page 106\)](#)

7.10.11 Read project related attributes

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)

Application

By using this function you can get project related attributes from the TIA Portal Openness API. The provided information contains project attributes, project history and products utilized by the project.

Project attributes

The project attributes provide the following information:

Attribute name	Data type	Writable	Description
Author	System.String	r/o	The author of the project
Comment	Siemens.Engineering.MultilingualText	r/o	The comment of the project
Copyright	System.String	r/o	The copyright statement of the project

Attribute name	Data type	Writable	Description
CreationTime	System.DateTime	r/o	The time when project has been created
Family	System.String	r/o	The family of the project
IsModified	System.Boolean	r/o	Returns true if the project has been modified
LanguageSettings	Siemens.Engineering.LanguageSettings	r/o	Handles project languages
LastModified	System.DateTime	r/o	The time when project was last modified
LastModifiedBy	System.String	r/o	Who made the last modification
Name	System.String	r/o	The name of the project
Path	System.IO.FileInfo	r/o	The absolute path of the project
Size	System.Int64	r/o	The size of the project in KB
Version	System.String	r/o	The version of the project

Modify the following program code to access project related attributes:

```
Project project = ...;
string author = project.Author;
string name = project.Name;
string path = project.Path;
DateTime creationTime = project.CreationTime;
DateTime modificationTime = project.LastModified;
string lastModifiedBy = project.LastModifiedBy;
string version = project.Version;
MultilingualText comment = project.Comment;
string copyright = project.Copyright;
string family = project.Family;
Int64 size = project.Size;
LanguageSettings languageSettings = project.LanguageSettings;
```

Modify the following program code to enumerate the project languages:

```
Project project = ...;
LanguageComposition languages = project.LanguageSettings.Languages;
foreach (Language language in languages)
{
    CultureInfo lang = language.Culture;
}
```

Modify the following program code to get comment text:

```
Project project = ...;
Language english =
project.LanguageSettings.ActiveLanguages.Find(CultureInfo.GetCultureInfo("en-US"));

MultilingualText projectComment = project.Comment;
MultilingualTextItem textItem = project.Comment.Items.Find(english);
string text = textItem.Text;
```

Project history

The project history is a composition of `HistoryEntry` objects, which contain the following information:

Attribute name	Data type	Writable	Description
Text	System.String	r/o	The event description
DateTime	System.DateTime	r/o	The time when the event was occurred

Modify the following program code to enumerate through `HistoryEntries` and access their attributes:

```
Project project = ...;
HistoryEntryComposition historyEntryComposition = project.HistoryEntries;
foreach (HistoryEntry historyEntry in historyEntryComposition)
{
    string entryText = historyEntry.Text;
    DateTime entryTime = historyEntry.DateTime;
}
```

Note

The text attribute of `HistoryEntry` contains a string in the same language as UI. If a TIA Portal Openness application is attached to a TIA Portal with no UI, the string is always in English

Used Products

The object `UsedProduct` includes the following information:

Attribute name	Data type	Writable	Description
Name	System.String	r/o	The name of the product used
Version	System.String	r/o	The version of the product

Modify the following program code to enumerate through `UsedProduct` and access the attributes.

```
Project project = ...;
UsedProductComposition usedProductComposition = project.UsedProducts;
foreach (UsedProduct usedProduct in usedProductComposition)
{
    string productName = usedProduct.Name;
    string productVersion = usedProduct.Version;
}
```

7.10.12 Deleting project graphics

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Program code

Modify the following program code to delete a project graphics:

```
//Deletes a single project graphic entry
public static void DeletesSingleProjectGraphicEntry(Project project)
{
    MultiLingualGraphicComposition graphicsAggregation = project.Graphics;
    MultiLingualGraphic graphic = graphicsAggregation.Find("Graphic XYZ");
    graphic.Delete();
}
```

7.10.13 Compiling a project

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- All devices are "Offline".

Application

The API interface supports the compilation of devices and program blocks. The compilation result is returned as an object. Depending on type of the object HW or SW or HW/SW compilation will be provided. The following object types are supported:

- Device - HW & SW
 - Device with failsafe CPU - SW with switched-off F-activation property
- DeviceItem - HW
- CodeBlock - SW
- DataBlock - SW
- HmiTarget - SW

7.10 Functions for projects and project data

- PlcSoftware – SW
 - PlcType – SW
 - PlcBlockSystemGroup – SW
 - PlcBlockUserGroup – SW
 - PlcTypeSystemGroup – SW
 - PlcTypeUserGroup – SW
-

Note**Time stamp format**

All time stamps are in UTC. If you want to see the local time you can use `DateTime.ToLocalTime()`.

Signature

Use the `ICompilable` method for compilation.

```
ICompilable compileService =
iEngineeringServiceProvider.GetService<ICompilable>();
CompilerResult result = compileService.Compile();
```

Note

All devices must be "Offline" before you start compiling.

Program code

Modify the following program code to compile the software changes of an object of the type `HmiTarget`:

```
public static void CompileHmiTarget(HmiTarget hmiTarget)
{
    ICompilable compileService = hmiTarget.GetService<ICompilable>();
    CompilerResult result = compileService.Compile();
}
```

Modify the following program code to compile the software changes of an object of the type `PlcSoftware`:

```
public static void CompilePlcSoftware(PlcSoftware plcSoftware)
{
    ICompilable compileService = plcSoftware.GetService<ICompilable>();
    CompilerResult result = compileService.Compile();
}
```

Modify the following program code to compile the software changes of an object of the type CodeBlock:

```
public static void CompileCodeBlock(PlcSoftware plcSoftware)
{
    CodeBlock block = plcSoftware.BlockGroup.Blocks.Find("MyCodeBlock") as CodeBlock;
    if (block != null)
    {
        ICompilable compileService = block.GetService<ICompilable>();
        CompilerResult result = compileService.Compile();
    }
}
```

Modify the following program code to evaluate the compilation result:

```
private void WriteCompilerResults(CompilerResult result)
{
    Console.WriteLine("State:" + result.State);
    Console.WriteLine("Warning Count:" + result.WarningCount);
    Console.WriteLine("Error Count:" + result.ErrorCount);
    RecursivelyWriteMessages(result.Messages);
}
private void RecursivelyWriteMessages(CompilerResultMessageComposition messages, string indent = "")
{
    indent += "\t";
    foreach (CompilerResultMessage message in messages)
    {
        Console.WriteLine(indent + "Path: " + message.Path);
        Console.WriteLine(indent + "DateTime: " + message.DateTime);
        Console.WriteLine(indent + "State: " + message.State);
        Console.WriteLine(indent + "Description: " + message.Description);
        Console.WriteLine(indent + "Warning Count: " + message.WarningCount);
        Console.WriteLine(indent + "Error Count: " + message.ErrorCount);
        RecursivelyWriteMessages(message.Messages, indent);
    }
}
```

7.10.14 Saving a project

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

To save a project

- Use the `Save()` method to save a project
- Use the `SaveAs()` method to save a project with a different name or in a different directory

Program code

Modify the following program code open and save a project:

```
public static void SaveProject(TiaPortal tiaPortal)
{
    Project project = null;
    //Use the code in the try block to open and save a project
    try
    {
        //please adapt the path and the extension apx to the installed version of TIA Portal
        project = tiaPortal.Projects.Open(new FileInfo(@"Some\Path\MyProject.apx"));
        //begin of code for further implementation
        //...
        //end of code
        project.Save();
    }
    //Use the code in the final block to close a project
    finally
    {
        if (project != null)
            project.Close();
    }
}
```

Modify the following program code save a project with a different name or in a different location:

```
...
    TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
    FileInfo fileInfoExistingProject = new FileInfo(@"D:\SampleProjects
\SampleProject.apXX");
    DirectoryInfo dirInfoSaveAsProject = new DirectoryInfo(@"D:\SampleProjects
\SampleProjectSaveAs");
    Project sampleProject = portal.Projects.Open(fileInfoExistingProject );
    sampleProject.SaveAs(dirInfoSaveAsProject);
...
}
```

7.10.15 Closing a project

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- You have opened a project with your TIA Portal Openness application.
See Opening a project (Page 106)

Program code

Modify the following program code to close a project:

```
public static void CloseProject(Project project)
{
    project.Close();
}
```

7.11 Functions for Connections

7.11.1 Configurable attributes of a port-to-port connection

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Opening a project (Page 106)
- A project is open.
See Opening a project (Page 106)

Application

The attributes of a port interconnection are located at the port device item. The read and write access of attributes via TIA Portal Openness is the same as at the UI.

Port interface settings

The following attributes are provided for port interface settings:

Attribute name	Data type	Writable	Access	Description
MediumAttachmentType	MediumAttachment-Type	r/o	Dynamic attribute	
CableName	CableName	r/w	Dynamic attribute	

7.11 Functions for Connections

Attribute name	Data type	Writable	Access	Description
AlternativePartnerPorts	Boolean	r/w	Dynamic attribute	Only available if toolchanger functionality is supported, e.g. at CPU1516.
SignalDelaySelection	SignalDelaySelection	r/w	Dynamic attribute	
CableLength	CableLength	r/w	Dynamic attribute	
SignalDelayTime	Double	r/w	Dynamic attribute	

The following ENUM values are provided for the attribute MediumAttachmentType:

Value	Description
MediumAttachmentType.None	Attachment type cannot be determined.
MediumAttachmentType.Copper	Attachment type is copper.
MediumAttachmentType.FibreOptic	Attachment type is fiber optic.

The following ENUM values are provided for the attribute CableName:

Value	Description
CableName.None	No cable name is specified
CableName.FO_Standard_Cable_9	FO standard cable GP (9 µm)
CableName.Flexible_FO_Cable_9	Flexible FO cable (9 µm)
CableName.FO_Standard_Cable_GP_50	FO standard cable GP (50 µm)
CableName.FO_Trailing_Cable_GP	FO trailing cable / GP
CableName.FO_Ground_Cable	FO ground cable
CableName.FO_Standard_Cable_62_5	FO standard cable (62.5 µm)
CableName.Flexible_FO_Cable_62_5	Flexible FO cable (62.5 µm)
CableName.POF_Standard_Cable_GP	POF standard cable GP
CableName.POF_Trailing_Cable	POF trailing cable
CableName.PCF_Standard_Cable_GP	PCF standard cable GP
CableName.PCF_Trailing_Cable_GP	PCF trailing cable / GP
CableName.GI_POF_Standard_Cable	GI-POF standard cable
CableName.GI_POF_Trailing_Cable	GI-POF trailing cable
CableName.GI_PCF_Standard_Cable	GI-PCF standard cable
CableName.GI_PCF_Trailing_Cable	GI-PCF trailing cable

The following ENUM values are provided for the attribute SignalDelaySelection:

Value	Description
SignalDelaySelection.None	
SignalDelaySelection.CableLength	CableLength is used to define the signal delay.
SignalDelaySelection.SignalDelayTime	SignalDelayTime is used to define the signal delay.

The following ENUM values are provided for the attribute `CableLength`:

Value	Description
<code>CableLength.None</code>	Cable length is not specified.
<code>CableLength.Length20m</code>	Cable length is 20m.
<code>CableLength.Length50m</code>	Cable length is 50m.
<code>CableLength.Length100m</code>	Cable length is 100m.
<code>CableLength.Length1000m</code>	Cable length is 1000m.
<code>CableLength.Length3000m</code>	Cable length is 3000m.

Port options

The following attributes are provided for port options:

Attribute name	Data type	Writable	Access
<code>PortActivation</code>	<code>bool</code>	r/w	Dynamic attribute
<code>TransmissionRateAndDuplex</code>	<code>TransmissionRateAndDuplex</code>	r/w	Dynamic attribute
<code>PortMonitoring</code>	<code>bool</code>	r/w	Dynamic attribute
<code>TransmissionRateAutoNegotiation</code>	<code>bool</code>	r/w	Dynamic attribute
<code>EndOfDetectionOfAccessibleDevices</code>	<code>bool</code>	r/w	Dynamic attribute
<code>EndOfTopologyDiscovery</code>	<code>bool</code>	r/w	Dynamic attribute
<code>EndOfSyncDomain</code>	<code>bool</code>	r/w	Dynamic attribute

The following ENUM values are provided for the attribute `TransmissionRateAndDuplex`:

Value	Description
<code>TransmissionRateAndDuplex.None</code>	
<code>TransmissionRateAndDuplex.Automatic</code>	Automatic
<code>TransmissionRateAndDuplex.AUI10Mbps</code>	10 Mbps AUI
<code>TransmissionRateAndDuplex.TP10MbpsHalfDuplex</code>	TP 10 Mbps half duplex
<code>TransmissionRateAndDuplex.TP10MbpsFullDuplex</code>	TP 10 Mbps full duplex
<code>TransmissionRateAndDuplex.AsyncFiber10MbpsHalfDuplex</code>	async fiber 10Mbit/s half duplex mode
<code>TransmissionRateAndDuplex.AsyncFiber10MbpsFullDuplex</code>	async fiber 10Mbit/s full duplex mode
<code>TransmissionRateAndDuplex.TP100MbpsHalfDuplex</code>	TP 100 Mbps half duplex
<code>TransmissionRateAndDuplex.TP100MbpsFullDuplex</code>	TP 100 Mbps full duplex
<code>TransmissionRateAndDuplex.FO100MbpsFullDuplex</code>	FO 100 Mbps full duplex
<code>TransmissionRateAndDuplex.X1000MbpsFullDuplex</code>	X1000 Mbps full Duplex

Value	Description
TransmissionRateAndDuplex.FO1000MbpsFullDuplexLD	FO 1000 Mbps full duplex LD
TransmissionRateAndDuplex.FO1000MbpsFullDuplex	FO 1000 Mbps full Duplex
TransmissionRateAndDuplex.TP1000MbpsFullDuplex	TP 1000 Mbps full duplex
TransmissionRateAndDuplex.FO10000MbpsFullDuplex	FO 10000 Mbps full Duplex
TransmissionRateAndDuplex.FO100MbpsFullDuplexLD	FO 100 Mbps full duplex LD
TransmissionRateAndDuplex.POFPCF100MbpsFullDuplex	POF/PCF 100 Mbps full duplex

See also

[Connecting to the TIA Portal \(Page 76\)](#)

7.12 Functions on libraries

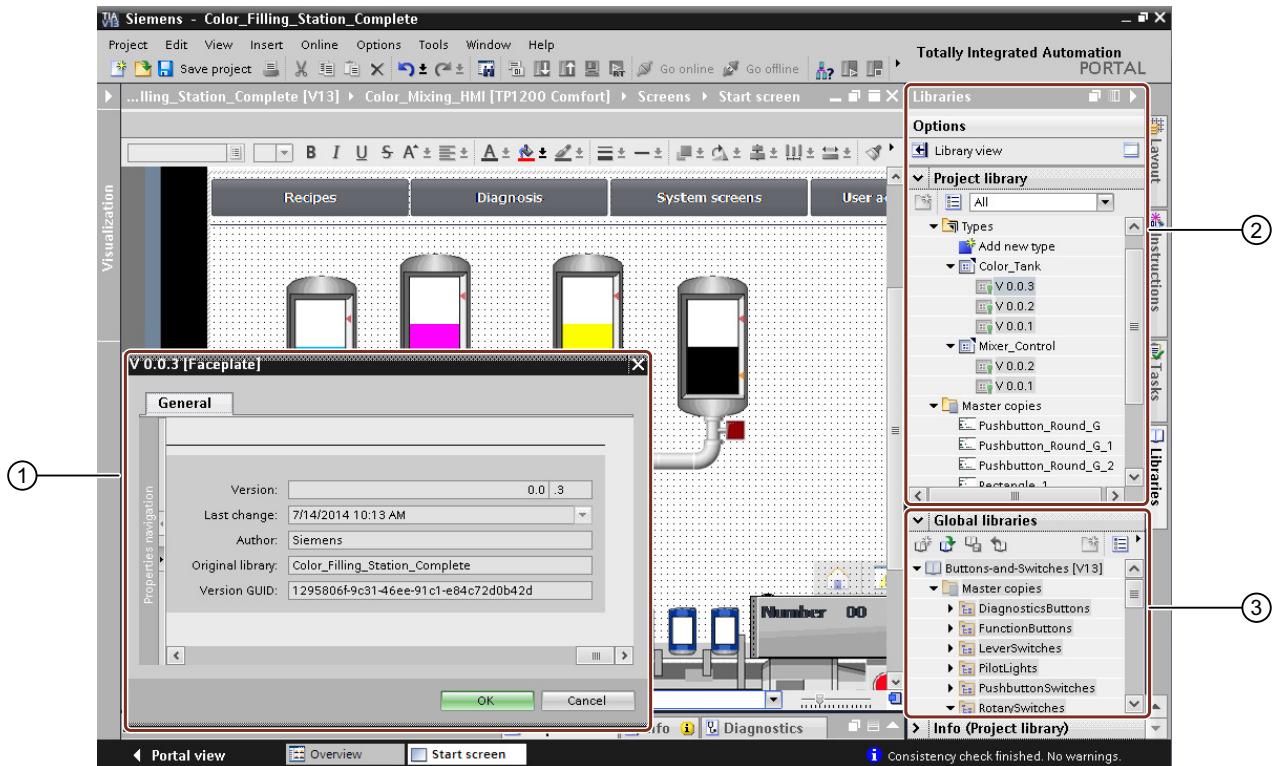
7.12.1 Functions for objects and instances

Accessing types and instances

You can use the TIA Portal Openness API interface to access types, type versions and master copies in the project library or global libraries. You can determine connections between type versions and instances. You can also update instances in the project and synchronize changes between a global library and the project library. The TIA Portal Openness API interface also supports the comparison of types versions and instances.

Functions for objects and instances

You have access to the following functions for types, type versions, master copies and instances with the TIA Portal Openness API interface:



- ① Display attributes of types, type versions, master copies and instances
- ② The following functions are available in the project library:
 - Update instances of types
 - Instantiating type versions in the project
 - Navigate within the library group
 - Delete groups, types, type versions and master copies
- ③ The following functions are available in the global library:
 - Update instances of types
 - Instantiate type version in the project
 - Navigate within the library group

7.12.2 Accessing global libraries

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)

Application

Three types of Global Libraries are existing.

- System global library (Siemens.Engineering.Library.SystemGlobalLibrary): These global libraries are included as part of a TIA Portal installation and use the .asx file extension. All system global libraries are read only.
- Corporate global library (Siemens.Engineering.Library.CorporateGlobalLibrary): These global libraries have been chosen by an administrator to be preloaded when TIA Portal is started. All corporate global libraries are read only.
- User global library (Siemens.Engineering.Library.UserGlobalLibrary): These global libraries have been created by users of TIA Portal. User global libraries can be opened either as read only mode or readwrite mode.
If an user global library is already opened in a certain mode then the same user global library cannot be opened with another mode.
User global libraries from previous versions can be opened only in read only mode.

Global Libraries opened using TIA Portal Openness will also be added to the TIA Portal UI's global library collection, and seen in the TIA Portal UI if the UI is present.

Program code: Available global libraries

Modify the following program code to get informations about all available global libraries:

```
TiaPortal tia = ...;
var availableLibraries = tia.GlobalLibraries.GetGlobalLibraryInfos();
foreach (GlobalLibraryInfo info in availableLibraries)
{
    Console.WriteLine(" Library Name: {0}", info.Name);
    Console.WriteLine(" Library Path: {0}", info.Path);
    Console.WriteLine(" Library Type: {0}", info.LibraryType);
    Console.WriteLine(" Library IsOpen: {0}", info.IsOpen);
}
```

Attributes of GlobalLibrary

Value	Data type	Description
Author	String	Author of the global library.
Comment	MultilingualText	Comment of the global library.
IsReadOnly	Boolean	True if the global library is read only.
IsModified	Boolean	True if the contents of the global library has been modified.
Name	String	Name of the global library.
Path	FileInfo	Path of the global library.

Attributes of GlobalLibraryInfo

Value	Return Type	Description
IsReadOnly	Boolean	True if the global library is read only.
IsOpen	Boolean	True if the global library is already open.
LibraryType	GlobalLibraryType	Type of the global library: • System: System global library • Corporate: Corporate global library • User: User global library
Name	String	Name of the global library.
Path	FileInfo	Path of the global library.

See also

[Accessing folders in a library \(Page 148\)](#)

7.12.3 Accessing global library languages

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
[See Connecting to the TIA Portal \(Page 76\)](#)
- A library is open
[See Opening libraries \(Page 141\)](#)

Application

You can use language setting navigator to access and manage the Global Library languages.

TIA Portal Openness supports the following access to the global library languages:

- Iterating through supported languages.
- Searching through the collection of supported languages by `System.Globalization.CultureInfo`.
- Accessing individual languages. Each `Language` object will contain a single read-only attribute `Culture` of type `System.Globalization.CultureInfo`.
- Accessing a collection of active languages.
- Searching through the collection of active languages by `System.Globalization.CultureInfo`.
- Adding a language to a collection of active languages.
- Removing a language from a collection of active languages.
- Setting an editing language.
- Setting a reference language.

Attribute of global library languages

Global library languages provides the following attribute:

Attribute name	Data type	Writeable	Description
LanguageSettings	Siemens.Engineering.LanguageSettings	r/o	Handles global library languages

Program code: Enumerating global library language

Modify the following program code to enumerate the global library language:

```
FileInfo m_GlobalLibrarypath = new FileInfo("bla");
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
var globalLibrary = portal.GlobalLibraries.Open(m_GlobalLibrarypath, OpenMode.ReadOnly);
LanguageSettings languageSettings = globalLibrary.LanguageSettings;
```

Program code: Accessing language setting

Modify the following program code to access the language setting on global library:

```
FileInfo m_GlobalLibrarypath = new FileInfo("bla");
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
var globalLibrary = portal.GlobalLibraries.Open(m_GlobalLibrarypath, OpenMode.ReadOnly);
LanguageComposition languages = globalLibrary.LanguageSettings.Languages;
foreach (Language language in languages)
{
    // Work with this language
}
```

Program code: Setting global library languages

Modify the following program code to set global library languages. If you set a new supported language through TIA Portal Openness, the language will be added to the active languages collection.

```
FileInfo m_GlobalLibrarypath = new FileInfo("bla");
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
var globalLibrary = portal.GlobalLibraries.Open(m_GlobalLibrarypath, OpenMode.ReadOnly);
LanguageSettings languageSettings = globalLibrary.LanguageSettings;
LanguageComposition supportedLanguages = languageSettings.Languages;
LanguageAssociation activeLanguages = languageSettings.ActiveLanguages;
Language supportedGermanLanguage = supportedLanguages.Find(CultureInfo.GetCultureInfo("de-DE"));
activeLanguages.Add(supportedGermanLanguage);
languageSettings.EditingLanguage = supportedGermanLanguage;
languageSettings.ReferenceLanguage = supportedGermanLanguage;
```

Note

Adding and modifying languages in global library language setting does not modify languages of released versions in the global library. This behaviour holds true for updating global library languages through UI (language editor) or LanguageSettings navigator.

7.12.4 Opening libraries

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- You have opened a project with your TIA Portal Openness application. This requirement is only for accessing project libraries.
See Opening a project (Page 106)

Application

A global library can be opened using System.IO.FileInfo with a path to the library file on a local storage medium or a network storage. Only user global libraries can be opened by path. A path obtained from a system global library or a corporate global library can not be used to open it.

As of V14 SP1 global libraries can be opened using the GlobalLibraryInfo. The OpenMode is specified in the GlobalLibraryInfo.

A user global library from a previous version of TIA Portal can be upgraded and opened with the current version of TIA Portal. A global library from V13 or a previous version cannot be opened with upgrade. These libraries have to be upgraded to V13 SP1 first.

Libraries opened using TIA Portal Openness will also be added to the global library collection in the TIA Portal, and will be visible in the user interface of TIA Portal.

Program code: Opening a library using System.IO.FileInfo

Modify the following program code:

```
TiaPortal tia = ...  
FileInfo fileInfo = ....  
  
UserGlobalLibrary userLib = tia.GlobalLibraries.Open(fileInfo, OpenMode.ReadWrite);
```

Program code: Opening a library using GlobalLibraryInfo

Modify the following program code:

```
TiaPortal tia = ...
IList<GlobalLibraryInfo> libraryInfos = tia.GlobalLibraries.GetGlobalLibraryInfos();
GlobalLibraryInfo libInfo = ...; //check for the info you need from the list, e.g.
GlobalLibrary libraryOpenedWithInfo;
if (libInfo.Name == "myLibrary")
libraryOpenedWithInfo = tia.GlobalLibraries.Open(libInfo);
```

Program code: Upgrading a library

Modify the following program code:

```
TiaPortal tia = ...
FileInfo fileInfo = .... //library from previous TIA Portal version

UserGlobalLibrary userLib = tia.GlobalLibraries.OpenWithUpgrade(fileInfo);
```

OpenMode

Value	Description
ReadOnly	Read access to the library.
ReadWrite	Read and write access to the library.

7.12.5 Enumerating open libraries**Requirement**

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)

Application

All opened global libraries in the TIA Portal, regardless if they have been opened via API or via user interface can be enumerated.

Global Libraries from previous versions of TIA Portal will not be enumerated if they are opened with write access.

Program code

Modify the following program code to enumerate open global libraries.

```
TiaPortal tia = ...  
foreach (GlobalLibrary globLib in tia.GlobalLibraries)  
{  
    ///work with the global library  
}
```

See also

[Opening a project \(Page 106\)](#)

7.12.6 Saving and closing libraries

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
[See Connecting to the TIA Portal \(Page 76\)](#)
- A library is open.
[See Opening libraries \(Page 141\)](#)

Application

User global libraries can be closed or saved. Any changes made to the global library will not be saved automatically. All unsaved changes will be discarded without prompting by closing a global library.

System global libraries and corporate global library cannot be closed or saved.

To save and close a global library:

- Use the Save () method to save a user global library
- Use the SaveAs () method to save a user global library in a different directory
- Use the Close () method to close a user global library

Feature tokens included for SaveAs () are:

- PublicAPI : Necessary for any Published feature
- DenyIfTransaction : Necessary since SaveAs should not be allowed within a transaction as it cannot be undone.

Program code

Modify the following program code to save a user global library:

```
UserGlobalLibrary userLib = ...  
// save changes and close library  
userLib.Save();  
userLib.Close();
```

Modify the following program code to save a user global library in a different location:

```
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);  
GlobalLibraryComposition globalLibraryComposition = portal.GlobalLibraries;  
//please adapt the path and the extension alx to the installed version of TIA Portal  
FileInfo existingLibraryfileInfo = new FileInfo(@"D:\GlobalLibraries\MyGlobalLibrary  
\MyGlobalLibrary.alx");  
DirectoryInfo targetDirectoryInfo = new DirectoryInfo(@"D:\GlobalLibraries  
\GlobalLibrarySaveAs");  
UserGlobalLibrary userGlobalLibrary =  
globalLibraryComposition.Open(existingLibraryfileInfo, OpenMode.ReadWrite);  
userGlobalLibrary.SaveAs(targetDirectoryInfo);
```

Note

The persistence of the library is changed after SaveAs operation. Thus, after the SaveAs is performed, the newly saved library is available in libraries card. However, the original library on which SaveAs operation was performed, would not be available. The newly saved library would have the same mode as the original library on which SaveAs was performed. This means if the existing library was opened in ReadOnly mode, then the newly saved library in the target path would also open in the ReadOnly mode. Similar is the case for ReadWrite mode.

Modify the following program code to close a user global library:

```
UserGlobalLibrary userLib = ...  
// close and discard changes  
userLib.Close();
```

7.12.7 Archiving and retrieving a library

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A library is open.
See Opening libraries (Page 141)
- A library is save
See Saving and closing libraries (Page 143)

Application

A opened and saved global libraries can be archived prior to any further modification to prevent from unintended result so that you can later retrieve the archived library. You can also share the archived file across the network easily.

Archiving a library

You can use the TIA Portal Openness API interface to archive a user global library. The API is available on the "Siemens.Engineering.UserGlobalLibrary" object.

```
public void Archive(System.IO.DirectoryInfo targetDirectory, string targetName,  
Siemens.Engineering.LibraryArchivationMode archivationMode)
```

'targetName' is the name of the file created for archived or non archived. This file may or may not contain any file extensions. If you do not provide any extension or provide an extension apart from "zalx" or "zal14" etc., then the archived file could not be retrieved from TIA Portal outside the Openness API.

For LibraryArchivationMode value as Compressed and

DiscardRestorableDataAndCompressed, the archived file name is same as it is provided by you. For LibraryArchivationMode value as None and DiscardRestorableData, the Library file extension is automatically decided by TIA Portal Project Manager component, based on the current version of TIA Portal.

Note

You must have saved the library before calling Archive API. In case the library contains any unsaved changes, archive would throw an EngineeringTargetInvocationException.

Library Archivation Mode

The LibraryArchivationMode enumeration have four values:

LibraryArchivation-Mode	Description
None	<ul style="list-style-type: none"> No special action are taken with the orginial files. Mode is similiar to a "save as" operation. No compressed zip file is created in this mode. The difference with SaveAs in this case is that the Archive will not change the persistence location to the new Archived folder whereas SaveAs does that.
DiscardRestorable-Data	<ul style="list-style-type: none"> The file storage stores the library in an internal data file and this file grows whenever a library data modification happens. In the case of DiscardRestorableData mode this data file is reorganized (only latest version of the objects are stored and history is removed from the file) and Intermediate data, the files of the IM directory and tmp directory (see Library Directory structure) are not copied to the archive location. No compressed zip file is created in this mode.
Compressed	The tmp library folder structure, created by Archiving, is compressed into a zip compatible archive. The tmp folder structure is removed after creation of the zip file.
DiscardRestorable-DataAndCom-pressed	The tmp library folder structure, created by Archiving, discards the restorable data and then compressed into a zip compatible archive. The tmp folder structure is removed after creation of the zip file.

Program code: Archiving a library

Modify the following program code to archive a user global library:

```
var tiaPortal = new TiaPortal(TiaPortalMode.WithoutUserInterface);
//Please adapt the path and the extension alx to the installed version of TIA Portal
var libraryFilePath = @"E:\Sample1\Sample1.alx";
var userGlobalLibrary = tiaPortal.GlobalLibraries.Open(new FileInfo(libraryFilePath),
OpenMode.ReadWrite);
var archivePath = @"E:\Archive";
var archiveFileName = "SampleArchive";
userGlobalLibrary.Archive(new DirectoryInfo(archivePath), archiveFileName,
LibraryArchivationMode.Compressed);
```

Retrieving a library

You can use the TIA Portal Openness API interface to retrieve an archived TIA Portal library. You can only retrieve a compressed archive. The API is available on the "Siemens.Engineering.GlobalLibraryComposition" object.

```
public Siemens.Engineering.Library.UserGlobalLibrary Retrieve(System.IO.FileInfo
sourcePath, System.IO.DirectoryInfo targetDirectory, Siemens.Engineering.OpenMode openMode)
```

Note

You cannot retrieve an archived library with 'LibraryArchivationMode.None' or 'LibraryArchivationMode.DiscardRestorableData' enumeration value.

You can call RetrieveWithUpgrade API for the archived library of a previous TIA Portal version. The API definition looks like the following:

```
public Siemens.Engineering.Library.UserGlobalLibrary
RetrieveWithUpgrade(System.IO.FileInfo sourcePath, System.IO.DirectoryInfo
targetDirectory, Siemens.Engineering.OpenMode openMode)
```

Open Mode

The OpenMode enumeration have two values:

OpenMode	Description
ReadMode	<ul style="list-style-type: none"> • Read access to the library. Data can be read from the library.
ReadWrite	<ul style="list-style-type: none"> • Write access to the library. Data can be written to the library.

Program code: Retrieving a library

Modify the following program code to access to retrieve a library:

```
//Please adapt the path and the extension zalx to the installed version of TIA Portal
var archivePath = @"E:\Archive\Sample1.zalx";
var retrievedLibraryDirectory = @"E:\RetrievedLibraries";
var tiaPortal = new TiaPortal(TiaPortalMode.WithoutUserInterface);
tiaPortal.GlobalLibraries.Retrieve(new FileInfo(archivePath), new
DirectoryInfo(retrievedLibraryDirectory), OpenMode.ReadWrite);
```

See also

[Opening libraries \(Page 141\)](#)

[Saving and closing libraries \(Page 143\)](#)

7.12.8 Creating global libraries

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)

Application

Global libraries can be created via TIA Portal Openness API by calling the Create method on the GlobalLibraryComposition. A UserGlobalLibrary will be returned

GlobalLibraryComposition.Create

Modify the following program code:

```
TiaPortal tia= ...;
DirectoryInfo targetDirectory = new DirectoryInfo(@"D:\GlobalLibraries");
UserGlobalLibrary globalLibrary =
tia.GlobalLibraries.Create<UserGlobalLibrary>(targetDirectory, "Library1")
```

According to this example

- a folder "D:\GlobalLibraries\Library1" will be created
- a global library file "D:\GlobalLibraries\Library1\Library1.aXXX" will be created

Parameters for creating global libraries

Parameter	Data Type	Type	Description
Author	String	Mandatory	Author of a global library.
Comment	String	Optional	Comment of a global library.
Name	String	Optional	Name of a global library
TargetDirectory	DirectoryInfo	Mandatory	Directory that will contain global library folder.

See also

[Opening a project \(Page 106\)](#)

7.12.9 Accessing folders in a library**Requirement**

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- You have opened a project with your TIA Portal Openness application.
See [Opening a project \(Page 106\)](#)
- You have access to the required library.
See [Accessing global libraries \(Page 137\)](#).

Application

You can use the TIA Portal Openness API interface to access the system folders for types and master copies in a library. You can access types, type versions, master copies and user-defined folders within the system folder.

You can access a user-defined folder at any time using the `Find` method, for example
`libTypeUserFolder.Folders.Find("SomeUserFolder");`.

Program code: Accessing system folders

Modify the following program code to access the system folder for types in a library:

```
public static void AccessTypeSystemFolder(ILibrary library)
{
    LibraryTypeSystemFolder libTypeSystemFolder = library.TypeFolder;
}
```

Modify the following program code to access the system folder for master copies in a library:

```
public static void AccessMasterCopySystemFolder(ILibrary library)
{
    MasterCopySystemFolder libMasterCopySystemFolder = library.MasterCopyFolder;
}
```

Program code: Accessing user-defined folders via `Find()` method

Modify the following program code:

```
...
LibraryTypeUserFolderComposition userFolderComposition = ...
LibraryTypeUserFolder userFolder = userFolderComposition.Find("Name of user folder");
...
```

Program code: Enumerating user defined folders

Modify the following program code to enumerate user-defined subfolders in a system folder for types:

```
public static void EnumerateUserFoldersInTypeSystemFolder(ILibrary library)
{
    // Enumerating user folders in type system folder:
    LibraryTypeSystemFolder libTypeSystemFolder = library.TypeFolder;
    foreach (LibraryTypeUserFolder libTypeUserFolder in libTypeSystemFolder.Folders)
    {
        //...
    }
}
```

Modify the following program code to enumerate user-defined subfolders in a system folder for master copies:

```
public static void EnumerateUserFoldersInMasterCopySystemFolder(ILibrary library)
{
    // Enumerating user folders in master copy system folder:
    MasterCopySystemFolder libMasterCopySystemFolder = library.MasterCopyFolder;
    foreach (MasterCopyUserFolder libMasterCopyUserFolder in
libMasterCopySystemFolder.Folders)
    {
        //..
    }
}
```

Modify the following program code to enumerate user-defined subfolders in a user-defined folder for types:

```
public static void EnumerateAllUserFolders(LibraryTypeUserFolder libUserFolder)
{
    foreach (LibraryTypeUserFolder libSubUserFolder in libUserFolder.Folders)
    {
        EnumerateAllUserFolders(libSubUserFolder);
    }
}
```

Modify the following program code to enumerate user-defined subfolders in a user-defined folder for master copies:

```
public static void EnumerateAllUserFolders(MasterCopyUserFolder libUserFolder)
{
    foreach (MasterCopyUserFolder libSubUserFolder in libUserFolder.Folders)
    {
        EnumerateAllUserFolders(libSubUserFolder);
    }
}
```

Program code: Creating user-defined folders

Modify the following program code to create a user-defined folder for types:

```
var typeFolderComposition = ProjectLibrary.TypeFolder.Folders;
var newTypeUserFolder = typeFolderComposition.Create("NewTypeUserFolder");
```

Modify the following program code to create a user-defined folder for master copies:

```
var masterCopyFolderComposition = projectProjectLibrary.MasterCopyFolder.Folders;
MasterCopyUserFolder newMasterCopyUserFolder =
masterCopyFolderComposition.Create("NewMasterCopyUserFolder");
```

Program code: Renaming user-defined folders

Modify the following program code to create a user-defined folder for types:

```
var typeUserFolder =  
project.ProjectLibrary.TypeFolder.Folders.Find("SampleTypeUserFolderName");  
typeUserFolder.Name = "NewTypeUserFolderName";
```

```
var typeUserFolder = ProjectLibrary.TypeFolder.Folders.Find("SampleTypeUserFolderName");  
typeUserFolder.SetAttributes(new[] {new KeyValuePair<string,object>("Name",  
"NewTypeUserFolderName")});
```

Modify the following program code to create a user-defined folder for master copies:

```
var masterCopyUserFolder =  
project.ProjectLibrary.MasterCopyFolder.Folders.Find("SampleMasterCopyUserFolderName");  
masterCopyUserFolder.Name = "NewMasterCopyUserFolderName";
```

```
var masterCopyUserFolder =  
ProjectLibrary.MasterCopyFolder.Folders.Find("SampleMasterCopyUserFolderName");  
masterCopyUserFolder.SetAttributes(new[] {new KeyValuePair<string,object>("Name",  
"NewMasterCopyUserFolderName")});
```

See also

[Accessing master copies \(Page 162\)](#)

7.12.10 Accessing types

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
[See Connecting to the TIA Portal \(Page 76\)](#)
- You have opened a project with your TIA Portal Openness application.
[See Opening a project \(Page 106\)](#)
- You have access to the required library.
[See Accessing global libraries \(Page 137\).](#)
- You have access to a group for types.
[See Accessing folders in a library \(Page 148\).](#)

Application

You can access the types contained in a library via the TIA Portal Openness API interface.

- You can enumerate the types.
- You can rename types.
- You can access the following attributes for every type:

Attribute	Data type	Description
Author	String	Returns the name of the author.
Comment	MultilingualText	Returns the comment.
Guid	Guid	Returns the GUID of the type. ¹
Name	String	Returns the name of the type. ²

¹ You can find an individual type in a library using this attribute. The search is recursive.

² You can find an individual type in a folder using this attribute. Subfolders are not included in the search. A type name is not unique. There can be several types with the same name in different groups. However, the type Guid is unique.

Subclasses for library type objects

With TIA Portal Openness API you can access library type objects via sub-classes. The following subclasses are existing:

- Siemens.Engineering.Hmi.Faceplate.FaceplateLibraryType
- Siemens.Engineering.Hmi.Faceplate.FaceplateLibraryTypeVersion
- Siemens.Engineering.Hmi.RuntimeScripting.VBScriptLibraryType
- Siemens.Engineering.Hmi.RuntimeScripting.VBScriptLibraryTypeVersion
- Siemens.Engineering.Hmi.RuntimeScripting.CScriptLibraryType
- Siemens.Engineering.Hmi.RuntimeScripting.CScriptLibraryTypeVersion
- Siemens.Engineering.Hmi.Screen.ScreenLibraryType
- Siemens.Engineering.Hmi.Screen.ScreenLibraryTypeVersion
- Siemens.Engineering.Hmi.Screen.StyleLibraryType
- Siemens.Engineering.Hmi.Screen.StyleLibraryTypeVersion
- Siemens.Engineering.Hmi.Screen.StyleSheetLibraryTypeVersion
- Siemens.Engineering.Hmi.Tag.HmiUdtLibraryType
- Siemens.Engineering.Hmi.Tag.HmiUdtLibraryTypeVersion
- Siemens.Engineering.SW.Blocks.CodeBlockLibraryType
- Siemens.Engineering.SW.Blocks.CodeBlockLibraryTypeVersion
- Siemens.Engineering.SW.Types.PlcTypeLibraryType
- Siemens.Engineering.SW.Types.PlcTypeLibraryTypeVersion

The following code is an example of how to use the library type sub-classes

```
ProjectLibrary library = project.ProjectLibrary;
VBScriptLibraryType vbScriptType = ...;
VBScriptLibraryType libraryTypeAsVbScript = libraryType as VBScriptLibraryType;
VBScriptLibraryTypeVersion libraryTypeVersionAsVbScript = libraryTypeVersion as
VBScriptLibraryTypeVersion
```

Program code

Modify the following program code to enumerate all types in the system folder of a library:

```
public static void EnumerateTypesInTypesSystemFolder (LibraryTypeSystemFolder
libraryTypeSystemFolder)
{
    foreach (LibraryType libraryType in libraryTypeSystemFolder.Types)
    {
        //...
    }
}
```

Modify the following program code to enumerate all types in a user-defined folder of a library:

```
public static void EnumerateTypesInTypesUserFolder (LibraryTypeUserFolder
libraryTypeUserGroup)
{
    foreach (LibraryType libraryType in libraryTypeUserGroup.Types)
    {
        //...
    }
}
```

Modify the following program code to access the attributes of a type:

```
public static void InspectPropertiesOfType (LibraryType libTypeObject)
{
    string typeAuthor = libTypeObject.Author;
    MultilingualText typeComment = libTypeObject.Comment;
    string typeName = libTypeObject.Name;
    Guid typeGUID = libTypeObject.Guid;
}
```

7.12 Functions on libraries

Modify the following program code to find an individual type by its name or GUID:

```
public static void FindTypeObjectInLibrary(ILibrary library)
{
    // Find type object by its GUID in a given library:
    System.Guid targetGuid = ...;
    LibraryType libTypeByGUID = library.FindType(targetGuid);
    // Find type object by its name in a given group:
    LibraryTypeFolder libTypeSystemFolder = library.TypeFolder;
    LibraryType libTypeByName = libTypeSystemFolder.Types.Find("myTypeObject");
}
```

Modify the following program code to rename a type:

```
// Setting the name attribute
var type = project.ProjectLibrary.TypeFolder.Types.Find("SampleTypeName");
type.Name = "NewTypeName";

//Setting the name attribute dynamically
var type = project.ProjectLibrary.TypeFolder.Types.Find("SampleTypeName");
type.SetAttributes(new[] {new KeyValuePair<string,object>("Name", "NewTypeName")});
```

7.12.11 Accessing type versions

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- You have opened a project with your TIA Portal Openness application.
See Opening a project (Page 106)
- You have access to the required library.
See Accessing global libraries (Page 137).
- You have access to a group for types.
See Accessing folders in a library (Page 148).

Application

You can access type versions via the TIA Portal Openness API interface.

- You can enumerate the type versions of a type.
- You can determine the type to which a type version belongs.
- You can enumerate the instances of a type version.
- You can create a new instance of a type version.
- You can navigate from an instance to its connected version object.
- You can access the following attributes for every type version:

Attribute	Data type	Description
Author	String	Returns the name of the author.
Comment	MultilingualText	Returns the comment.
Guid	Guid	Returns the GUID of the type version. ¹
ModifiedDate	DateTime	Returns the date and time at which the type version was set to the "Committed" status.
State	LibraryTypeVersion-State	Returns the status of the version: <ul style="list-style-type: none"> • InWork: Corresponds to the status "In progress" or "In testing" depending on the associated type. • Committed: Corresponds to the status "Released".
TypeObject	LibraryType	Returns the type to which this type version belongs.
VersionNumber	Version	Returns the version number as a three digit version identification, for example, "1.0.0". ²

¹ You can find an individual type version in a library using this attribute.

² You can find an individual type version in a "LibraryTypeVersion" composition using this attribute.

Enumerate all type versions of a type

Modify the following program code:

```
//Enumerate the type versions of a type
public static void EnumerateVersionsInType(LibraryType libraryType)
{
    foreach (LibraryTypeVersion libraryTypeVersion in libraryType.Versions)
    {
        //...
    }
}
```

Accessing the attributes of a type version

Modify the following program code:

```
//Acessing the attributes of a type version
public static void InspectPropertiesOfVersion(LibraryTypeVersion libTypeVersion)
{
    string versionAuthor = libTypeVersion.Author;
    MultilingualText versionComment = libTypeVersion.Comment;
    Guid versionGUID = libTypeVersion.Guid; DateTime versionModifiedDate =
libTypeVersion.ModifiedDate;
    LibraryTypeVersionState versionStateLibrary = libTypeVersion.State;
    LibraryType versionParentObject = libTypeVersion.TypeObject;
    Version versionNumber = libTypeVersion.VersionNumber;
}
```

Creating an instance of a type version

You can create a new instance of a type version. The following objects are supported:

- Blocks (FB/FC)
- PLC user data types
- Screens
- VB scripts

An instance can be created of a type version from global library and project library. When you create the instance of a type version from a global library, the type version is first synchronized with the project library.

A recoverable Exception will be thrown if an instance cannot be created in the target, then .
Possible reasons are:

- The library type version is in-work
- An instance of the library type version already exists in the target device

Modify the following program code:

```
VBScriptLibraryTypeVersion scriptVersion = ...;  
VBScriptComposition vbscripts = ...;  
  
//Using the CreateFrom method to create an instance of the version in the VBScripts  
composition  
VBScript newScript = vbscripts.CreateFrom(scriptVersion);
```

Modify the following program code:

```
ScreenLibraryTypeVersion screenVersion = ...;  
ScreenComposition screens = ...;  
  
//Using the CreateFrom method to create an instance of the version in the screens  
composition  
Screen newScreen = screens.CreateFrom(screenVersion);
```

Modify the following program code:

```
CodeBlockLibraryTypeVersion blockVersion = ...;  
PlcBlockComposition blocks = ...;  
  
//Using the CreateFrom method to create an instance of the version in the blocks composition  
PlcBlock newBlock = blocks.CreateFrom(blockVersion);
```

Modify the following program code:

```
PlcTypeLibraryTypeVersion plcTypVersione=...;
PlcTypeComposition types=...;

//Using the CreateFrom method to create an instance of the version in the types composition
PlcType newType = types.CreateFrom(plcTypeVersion);
```

Determining uses of a type version

The following uses are distinguished for type versions:

- The type version uses other type versions from the library.
Example: A user data type is used in a program block. The program block must have access to the user data type. This means the program block depends on the user data type.
When you access the Dependencies attribute of CodeBlockLibraryVersion through GetDependencies() method, a list of LibraryTypeVersions are returned.
- The type is being used by another type version in the library.
Example: A user data type is used in a program block. The program block must have access to the user data type. The user data type has the associated program block. The program block depends on the user data type.
When you access the Dependents attribute of PlcTypeLibraryTypeVersion through GetDependents() method, a list of LibraryTypeVersions are returned.

Both attributes return a list that contains objects of the LibraryTypeVersion type. If there are no uses, an empty list is returned.

Note

If you use these attributes on type versions with the "InWork" status, an exception can be thrown.

Modify the following program code:

```
//Determine the uses of a type version in a library
public static void GetDependenciesAndDependentsOfAVersion(LibraryTypeVersion
libTypeVersion)
{
    IList<LibraryTypeVersion> versionDependents = libTypeVersion.Dependents();
    IList<LibraryTypeVersion> versionDependencies = libTypeVersion.Dependencies();
}
```

Program code

Modify the following program code to determine the type to which a type version belongs:

```
public static void GetParentTypeOfVersion(LibraryTypeVersion libTypeVersion)
{
    LibraryType parentType = libTypeVersion.TypeObject;
}
```

Modify the following program code to determine the master copies that contain instances of a type version:

```
public static void GetMasterCopiesContainingInstances(LibraryTypeVersion libTypeVersion)
{
    MasterCopyAssociation masterCopies = libTypeVersion.MasterCopiesContainingInstances;
}
```

Modify the following program code to find an individual type version by its version number:

```
public static void FindVersionInLibrary(ILibrary library, Guid versionGUID)
{
    LibraryTypeVersion libTypeVersionByVersionNumber = library.FindVersion(versionGUID);
}
```

7.12.12 Accessing blocks located in library

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- Opening a project
See Opening a project (Page 106)

Application

You can use the TIA Portal Openness to get version information from a library object without instantiating and compiling the object.

The following new Export Action will be provided on the LibraryTypeVersion engineering object to export the version content.

The action will create the Exported xml file at given exportFileInfo.

```
void Export(''FileInfo'' exportFileInfo, ''ExportOptions'' exportOptions)
```

Exception will occur:

- In case of the user exceptions stating "The version data cannot be exported as it is in-work state" and the version to be exported is in "in test" state
- In case of the General user DataExchange for example "FileAlreadyExists" exception stating "The export cannot be made because the file 'D:*.xml' already exists".

Library Type Version engineering Object

The following additions are made to the Library TypeVersion engineering Object:

1. Export action is provided on LibraryTypeVersion
2. Navigator to navigate to the version content browsable:
 - Navigator name : ContentObject
 - ReadPublicationLevel : System
 - Relation name: Engineering.Library.DefaultVersionContentObject
 - Base navigator will provide a default implementation to provide empty browsable collection as version content. Client can override this navigator and provide versioncontent.
3. "LibraryTypeName" and "LibraryTypeGuid" on LibraryTypeVersion object

Attributes	ReadPublicationLevel
LibraryTypeName	System
LibraryTypeGuid	System

Note

This is necessary because without this info you would not be able to understand that exported version info (in the exported xml file) belongs to which type.

Exported Content

Following content are available in the exported file:

- Library Version Info exported

Attributes	SimaticMLAAccess
Author	ReadWrite
Guid	ReadOnly
Modified Data	ReadOnly
VersionNumber	ReadWrite
LibraryTypeName	ReadOnly
Library TypeGuid	ReadOnly

Navigators Exported:

- Comment

Note

Attributes with SimaticMLAccess readonly get exported only if you choose the exportoption as "ExportOptions.WithReadOnly".

Program code

Modify the following program code to export version content information:

```
Guid g = new Guid("35ad9996-d6d7-40c9-989f-0f0c7e21a7b2");
//Block1
var version = m_Project.ProjectLibrary.FindType(g).Versions[0];
version.Export(new FileInfo(@"D:\ExportCodeBlock.xml"), ExportOptions.WithReadOnly);
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.12.13 Accessing instances

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- You have opened a project with your TIA Portal Openness application.
See [Opening a project \(Page 106\)](#)
- You have access to the required library.
See [Accessing global libraries \(Page 137\)](#).
- You have access to a group for types.
See [Accessing folders in a library \(Page 148\)](#).

Application

You can access instances of type versions via the TIA Portal Openness API interface.

Use the `FindInstances(IInstanceSearchScope searchScope)` method to find all instances of a type version.

You can use the `searchScope` parameter to specify the area of the project to be searched. The following classes implement the `IInstanceSearchScope` interface and can be used to search for instances:

- PlcSoftware
- HmiTarget

The method returns a list that contains objects of the `LibraryTypeInstanceStateInfo` type. If there are no instances, an empty list is returned.

Note

Instances of faceplates and HMI user data types are always linked to the associated type version.

Instances of all other objects, such as program blocks or screens, can be linked to a type version.

Enumerate the instances of a type version

Modify the following program code:

```
//Enumerate the instances of a type version in the project
LibraryTypeVersion version = ...;
PlcSoftware plcSoftware = ...;

IInstanceSearchScope searchScope = plcSoftware as IInstanceSearchScope;

if(searchScope==null)
{
    //No search possible
}

IList<LibraryTypeInstanceStateInfo> instanceInfos = version.FindInstances(searchScope);
IEnumerable<IEngineeringObject> instances = instanceInfos.Select(instanceInfo =>
instanceInfo.LibraryTypeInstance);
```

Navigate from an instance to its connected version object

Use the `LibraryTypeVersion` attribute of `LibraryTypeInstanceStateInfo` service to navigate from an instance to its connected version object.

The following objects provide the `LibraryTypeInstanceStateInfo` service:

- Blocks FB
- Blocks FC
- PLC user data types
- Screens
- VB scripts

7.12 Functions on libraries

If an instance object is not connected to a version object, then it will not provide the "LibraryTypeInstanceStateInfo" service.

```
FC fc = ...;
//Using LibraryTypeInstanceStateInfo service

LibraryTypeInstanceStateInfo instanceInfo = fc.GetService<LibraryTypeInstanceStateInfo>();
if(instanceInfo != null)
{
    LibraryTypeVersion connectedVersion = instanceInfo.LibraryTypeVersion;
    FC parentFc = instanceInfo.LibraryTypeInstance as FC; //parentFc == fc
}
```

7.12.14 Accessing master copies

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- You have access to the required library.
See Accessing global libraries (Page 137)
- You have access to a group for master copies.
See Accessing folders in a library (Page 148)

Application

The TIA Portal Openness API interface supports access to master copies in a global library and the project library:

- Creating master copies
- Enumerating master copies in system folders and user-defined folders
- Renaming master copies
- Querying information from master copies
- Querying information from objects in a master copy

Attribute	Data type	Description
Author	String	Returns the name of the author.
ContentDescriptions	MasterCopyContentDescriptionComposition	Returns a description for the content of the MasterCopy.
CreationDate	DateTime	Returns the creation date.
Name	String	Returns the name of the master copy.

Program code

Modify the following program code to enumerate all master copies in the system folder of a library:

```
public static void EnumerateMasterCopiesInSystemFolder
(MasterCopySystemFolder masterCopySystemFolder)
{
    foreach (MasterCopy masterCopy in masterCopySystemFolder.MasterCopies)
    {
        //...
    }
}
```

Modify the following program code to access an individual mastercopy by using the find method:

```
...
MasterCopySystemFolder systemFolder = projectLibrary.MasterCopyFolder;
MasterCopyComposition mastercopies = systemFolder.MasterCopies;
MasterCopy masterCopy = mastercopies.Find("Copy of ...");
...
```

Modify the following program code to enumerate master copy groups and subgroups:

```
private static void EnumerateFolder(MasterCopyFolder folder)
{
    EnumerateMasterCopies(folder.MasterCopies);
    foreach (MasterCopyUserFolder subFolder in folder.Folders)
    {
        EnumerateFolder(subFolder); // recursion
    }
}
private static void EnumerateMasterCopies(MasterCopyComposition masterCopies)
{
    foreach (MasterCopy masterCopy in masterCopies)
    {
        ...
    }
}
```

Modify the following program code to access a MasterCopyUserFolder by using the find method:

```
...
MasterCopyUserFolderComposition userFolderComposition = ...
MasterCopyUserFolder userFolder = userFolderComposition.Find("Name of user folder");
...
```

Modify the following program code to rename a master copy:

```
//Setting the name attribute
var masterCopy = projectLibrary.MasterCopyFolder.MasterCopies.Find("SampleMasterCopyName");
masterCopy.Name = "NewMasterCopyName";

//Setting the name attribute dynamically
var masterCopy = projectLibrary.MasterCopyFolder.MasterCopies.Find("SampleMasterCopyName");
masterCopy.SetAttributes(new[] {new KeyValuePair<string,object>("Name",
"NewMasterCopyName")});
```

Querying information from master copies

Modify the following program code to get information of a master copy:

```
public static void GetMasterCopyInformation(MasterCopy masterCopy)
{
    string author = masterCopy.Author;
    DateTime creationDate = masterCopy.CreationDate;
    string name = masterCopy.Name;
}
```

Querying information from objects in a master copy

The MasterCopy object contains a navigator called ContentDescriptions, which is a composition of MasterCopyContentDescriptions.

A master copy may contain multiple objects. The MasterCopy object contains ContentDescriptions for each object directly contained in the MasterCopy. If the master copy contains a folder which also contains some items, the MasterCopy object only contains one ContentDescription of the folder.

```
MasterCopy multiObjectMasterCopy = ...;

//Using ContentDescriptions
MasterCopyContentDescriptionComposition masterCopyContentDescriptions =
multiObjectMasterCopy.ContentDescriptions;
MasterCopyContentDescription contentDescription= masterCopyContentDescriptions.First();

string name = contentDescription.ContentName;
Type type = contentDescription.ContentType;
```

7.12.15 Create master copy from a project in library

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

If the library is a read-write library, you can create a MasterCopy of an **IMasterCopySource** at target location.

MasterCopy

```
MasterCopyComposition.Create(Siemens.Engineering.Library.MasterCopies.IMasterCopySource  
sourceObject);
```

An **EngineeringException** will be thrown if:

- The target location is read-only
- Creation of the MasterCopy from the source is rejected by the system

The following items are defined as **IMasterCopySources**:

- Device - HW
- DeviceItem - HW
- DeviceUserGroup - HW
- CodeBlock - SW
- DataBlock - SW
- PlcBlockUserGroup - SW
- PlcTag - SW
- PlcTagTable - SW
- PlcTagTableUserGroup - SW
- PlcType - SW
- PlcTypeUserGroup - SW
- VBScript - HMI
- VBScriptUserFolder - HMI
- Screen - HMI
- ScreenTemplate - HMI
- ScreenTemplateUserFolder - HMI
- ScreenUserFolder - HMI

- Tag - HMI
- TagTable - HMI
- TagUserFolder - HMI

Program code

Use the following program code:

```
// create a master copy from a code block in the project library
public static void Create(Project project, PlcSoftware plcSoftware)
{
    MasterCopySystemFolder masterCopyFolder = project.ProjectLibrary.MasterCopyFolder;
    CodeBlock block = plcSoftware.BlockGroup.Groups[0].Blocks.Find("Block_1") as CodeBlock;
    MasterCopy masterCopy = masterCopyFolder.MasterCopies.Create(block);
}
```

7.12.16 Create an object from a master copy

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- PLC is not online.

Application

The TIA Portal Openness API interface supports the use of master copies in the project. You can create an object in the object's composition from a master copy in a project library or a global library using the CreateFrom method.

The return type will correspond to the respective composition's return type.

The CreateFrom method only supports master copies containing single objects. If the composition where the action is called and the source master copy are incompatible (e.g. source master copy contains a plc tag table and the composition is a plc block composition), a recoverable exception will be thrown.

The following compositions are supported:

- Siemens.Engineering.HW.DeviceComposition
- Siemens.Engineering.HW.DeviceItemComposition
- Siemens.Engineering.SW.Blocks.PlcBlockComposition
- Siemens.Engineering.SW.Tags.PlcTagTableComposition
- Siemens.Engineering.SW.Tags.PlcTagComposition

- Siemens.Engineering.SW.Types.PlcTypeComposition
- Siemens.Engineering.SW.TechnologicalObjects.TechnologicalInstanceDBComposition
- Siemens.Engineering.SW.Tags.PlcUserConstantComposition
- Siemens.Engineering.Hmi.Tag.TagTableComposition
- Siemens.Engineering.Hmi.Tag.TagComposition
- Siemens.Engineering.Hmi.Screen.ScreenComposition
- Siemens.Engineering.Hmi.Screen.ScreenTemplateComposition
- Siemens.Engineering.Hmi.RuntimeScripting.VBScriptComposition
- Siemens.Engineering.HW.SubnetComposition
- Siemens.Engineering.HW.DeviceUserGroupComposition
- Siemens.Engineering.SW.Blocks.PlcBlockUserGroupComposition
- Siemens.Engineering.SW.ExternalSources.PlcExternalSourceUserGroupComposition
- Siemens.Engineering.SW.Tags.PlcTagTableUserGroupComposition
- Siemens.Engineering.SW.Types.PlcTypeUserGroupComposition

Program code: Create a PLC block from a mastercopy

Modify the following program code to create an PLC block from a master copy in a library:

```
var plcSoftware = ...;
MasterCopy copyOfPlcBlock = ...;
PlcBlock plcSoftware.BlockGroup.Blocks.CreateFrom(copyOfPlcBlock);
```

Program code: Create a device from a mastercopy

Modify the following program code to create a device from a master copy in a library:

```
Project project = ...;
MasterCopy copyOfDevice = ...;
Device newDevice = project.Devices.CreateFrom(copyOfDevice);
```

Program code: Create a device item from a mastercopy

Modify the following program code to create a device item from a master copy in a library:

```
Device device = ...;
MasterCopy copyOfDeviceItem = ...;
DeviceItem newDeviceItem = device.DeviceItems.CreateFrom(copyOfDeviceItem);
```

Program code: Create a subnet from a mastercopy

Modify the following program code to create a subnet from a master copy in a library:

```
Project project = ...;
MasterCopy copyOfSubnet = ...;
Subnet newSubnet = project.Subnets.CreateFrom(copyOfSubnet);
```

Program code: Create a device folder from a mastercopy

Modify the following program code to create a device folder from a master copy in a library:

```
Project project = ...;
MasterCopy copyOfDeviceGroup = ...;
DeviceGroup newDeviceGroup= project.DeviceGroups.CreateFrom(copyOfDeviceGroup);
```

See also

[Accessing master copies \(Page 162\)](#)

7.12.17 Copying master copies

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)

Application

The TIA Portal Openness API interface supports copying of master copies within a library and between libraries using the `CreateFrom` action. The action will create a new object based on the source master copy and place it in the composition where the action was called. The action will try to create the new master copy with the same name as the source master copy. If such name is not available, the system will give the new master copy a new name. Then, it will return the new master copy.

If the composition where the "CreateFrom" action is called is in a read-only global library, a recoverable exception will be thrown.

Program code

Modify the following program code:

```
ProjectLibrary projectLibrary = ...;

MasterCopy copiedMasterCopy =
projectLibrary.MasterCopyFolder.MasterCopies.CreateFrom(sampleMasterCopy)
```

See also

[Accessing master copies \(Page 162\)](#)

7.12.18 Determining out-of-date type instances

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)
- You have access to the required library.
See [Accessing global libraries \(Page 137\)](#)
- You have access to a folder for types.
See [Accessing folders in a library \(Page 148\)](#).

Application

The TIA Portal Openness API interface allows you to determine type versions which belong to the instances in the open project. The TIA Portal Openness API returns one of the following two states per instance:

- The instance refers to an out-of-date type version.
- The instance refers to the latest type version.

The following rules apply when determining the version:

- You determine the version based on a library and the project that you want to open via the TIA Portal Openness API interface.
- Instances are not updated when you determine the version.

Signature

Use the `UpdateCheck` method to determine the instances of a type version:
`UpdateCheck(Project project, UpdateCheckMode updateCheckMode)`

Parameter	Function
Project	Specifies the project in which the type versions of instances are determined.
UpdateCheckMode	Specifies the versions that are determined: <ul style="list-style-type: none"> ReportOutOfDateOnly: Returns only status of the "out of date" type. ReportOutOfDateAndUpToDate: Returns status of the type "out of date" and "up to date".

Result

The devices of the project are scanned from top to bottom when determining the version. Each device is checked to determine whether its configuration data contain an instance of a type version from the specified library. The `UpdateCheck` method returns the result of the version check in hierarchical order.

The table below shows a result of a version check with the parameter `UpdateCheck.ReportOutOfDateAndUpToDate`:

Update check for: HMI_1		
Update check for library element Screen_1 0.0.3		
Out-of-date		
\HMI_1\Screens		Screen_4 0.0.1
\HMI_1\Screens		Screen_2 0.0.2
Up-to-date		
\HMI_1\Screens		Screen_1 0.0.3
\HMI_1\Screens		Screen_10 0.0.3
Update check for: HMI_2		
Update check of library element Screen_4 0.0.3		
Out-of-date		
\Screens folder1		Screen_02 0.0.1
\Screens folder1		Screen_07 0.0.2
Up-to-date		
\Screens folder1		Screen_05 0.0.3
\Screens folder1		Screen_08 0.0.3

Program code

There is no difference between project and global libraries in handling the update check.

Modify the following program code to determine the type versions from a global or project library for instances in the project:

```
public static void UpdateCheckOfGlobalLibrary(Project project, ILibrary library)
{
    // check for out of date instances and report only out of date instances in the returned
    feedback
    UpdateCheckResult result = library.UpdateCheck(project,
UpdateCheckMode.ReportOutOfDateOnly);

    //Alternatively, check for out of date instances and report both out of date and up to
    date instances in the returned feedback
    UpdateCheckResult alternateResult = library.UpdateCheck(project,
UpdateCheckMode.ReportOutOfDateAndUpToDate);

    //Show result
    RecursivelyWriteMessages(result.Messages);

    // Alternatively, show result and access single message parts
    RecursivelyWriteMessageParts(result.Messages);
}
```

Modify the following program code to output the result of the version check and process the messages individually:

```
private static void RecursivelyWriteMessages (UpdateCheckResultMessageComposition
messages, string indent = "")
{
    indent += "\t";
    foreach (UpdateCheckResultMessage message in messages)
    {
        Console.WriteLine(indent + message.Description);
        RecursivelyWriteMessages(message.Messages, indent);
    }
}
```

Modify the following program code to access individual message parts in the result of the version check:

```
private static void RecursivelyWriteMessageParts (UpdateCheckResultMessageComposition messages, string indent= "")
{
    indent += "\t";
    foreach (UpdateCheckResultMessage message in messages)
    {
        Console.WriteLine(indent + "Full description: " + message.Description);
        foreach (KeyValuePair<string, string> messagePart in message.MessageParts)
        {
            // first level
            // part 1: device name
            // second level:
            // part 1: Name of the type in the global library
            // part 2: version of the type in the global library
            // third level:
            // part 1: title (either "Out-of-date" or "Up-to-date");
            // fourth level:
            // part 1: Path hierarchy to instance
            // part 2: Instance name in project
            // part 3: Version of the instance in the project
            Console.WriteLine(indent + "*Key: {0} Value:{1}", messagePart.Key,
messagePart.Value);
        }
        RecursivelyWriteMessageParts(message.Messages, indent);
    }
}
```

7.12.19 Updating the project

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- You have opened a project with your TIA Portal Openness application.
See Opening a project (Page 106)
- You have access to the required library.
See Accessing global libraries (Page 137).
- You have access to a folder for types.
See Accessing folders in a library (Page 148).

Application

The TIA Portal Openness API interface allows you to update instances of a selected type within a type folder in a project.

When updating, the instances used in the project are updated based on the last released type version. If you start updating the instances from a global library, synchronization is performed beforehand.

Signature

Use the `UpdateProject` method to update instances.

Use the following call for classes which implement the `LibraryTypes` interface:

```
void UpdateProject(IUpdateProjectScope updateProjectScope)
```

Use the following call for classes which implement the `ILibrary` interface:

```
void UpdateProject(IEnumerable<ILibraryTypeOrFolderSelection>
selectedTypesOrFolders, IEnumerable <IUpdateProjectScope>
updateProjectScope)
```

Each call is entered in the log file in the project directory.

Parameter	Function
<code>IEnumerable<ILibraryTypeOrFolderSelection> selectedTypesOrFolders</code>	Specifies the folder or types to be synchronized or their instances in the project to be updated.
<code>IUpdateProjectScope updateProjectScope</code> <code>IEnumerable <IUpdateProjectScope> updateProjectScope</code>	Specifies the object(s) in the project in which the uses of instances are to be updated. The following objects are supported: <ul style="list-style-type: none"> • PlcSoftware • HmiTarget

Program code

Modify the following program code to update instances of selected types within a type folder:

```
private static void UpdateInstances(ILibrary myLibrary, LibraryTypeFolder
singleFolderContainingTypes, LibraryType singleType, PlcSoftware plcSoftware, HmiTarget
hmiTarget)
{
    //Update Instances of multiple types (subset of types and folders)
    IUpdateProjectScope[] updateProjectScopes =
    {
        plcSoftware as IUpdateProjectScope, hmiTarget as IUpdateProjectScope
    };
    myLibrary.UpdateProject(new ILibraryTypeOrFolderSelection[] {singleType,
singleFolderContainingTypes}, updateProjectScopes);
    //Update Instances of multiple types (all types in library)
    myLibrary.UpdateProject(new[] {myLibrary.TypeFolder}, updateProjectScopes);
}
```

7.12.20 Updating a library

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- You have opened a project with your TIA Portal Openness application.
See Opening a project (Page 106)
- You have access to the required library.
See Accessing global libraries (Page 137).
- You have access to a folder for types.
See Accessing folders in a library (Page 148).

Application

The TIA Portal Openness API interface supports the following updates in the project library:

- Synchronize selected types between libraries.

The folder structure is not adapted when you perform synchronization. The types to be updated are identified by their GUID and updated:

- If a type in a library includes a type version that is missing in the library to be updated, the type version is copied.
- If a type in a library includes a type version with the different GUID, the process is aborted and an Exception is thrown.

Signature

Use the `UpdateLibrary` method to synchronize type versions.

Use the following call for classes which implement the `LibraryTypes` interface:

```
void UpdateLibrary(ILibrary targetLibrary)
```

Use the following call for classes which implement the `ILibrary` interface:

```
void UpdateLibrary(IEnumerable<LibraryTypeOrFolderSelection>
selectedTypesOrFolders, ILibrary targetLibrary)
```

Parameter	Function
<code>IEnumerable<ILibraryTypeOrFolderSelection> selectedTypesOrFolders</code>	Specifies the folder or types to be synchronized or their instances in the project to be updated.
<code>ILibrary targetLibrary</code>	Specifies the library whose contents will be synchronized with a library. If source library and destination library are identical, an exception is thrown.

Program code

Modify the following program code to synchronize a type from the project library with a global library:

```
sourceType.UpdateLibrary(projectLibrary);
```

Modify the following program code to synchronize selected types within a type folder between a global library and the project library:

```
globalLibrary.UpdateLibrary(new[] {globalLibrary.TypeFolder}, projectLibrary);
```

7.12.21 Deleting library content

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- You have opened a project with your TIA Portal Openness application.
See Opening a project (Page 106)
- You have access to the required library.
See Accessing global libraries (Page 137).
- You have access to a folder for types.
See Accessing folders in a library (Page 148).

Application

You can delete the following project library content using the TIA Portal Openness API interface:

- Types
- Type version
- User-defined folders for types
- Master copies
- User-defined folders for master copies

Note

Deleting of types and user-defined type folders

If you want to delete a type or user-defined folder type, the "Rules for deleting versions" must be met. You can always delete an empty type folder.

Note**Rules for deleting versions**

You can only delete versions with "Committed" status. The following rules also apply when deleting versions:

- If a new version with the "InWork" status has just been created from a version with "Committed" status , you can only delete the version with "Committed" status when the new version is discarded or it obtains the "Committed" status.
 - If a type only has one version, the type is deleted as well.
 - If Version A is dependent on Version B of another type, first delete Version A and then Version B.
 - If there are instances of Version A, you can only delete Version A if the instances are deleted as well. If an instance is also contained in a master copy, the master copy is deleted as well.
-

Program code

Modify the following program code to delete types or user-defined type folders:

```
public static void DeleteMultipleTypesOrTypeUserFolders(ILibrary library)
{
    LibraryType t1 = library.TypeFolder.Types.Find("type1");
    LibraryType t2 = library.TypeFolder.Types.Find("type2");
    LibraryTypeUserFolder f1 = library.TypeFolder.Folders.Find("folder1");
    t1.Delete();
    t2.Delete();
    f1.Delete();
}
```

Modify the following program code to delete an individual type or user-defined type folder:

```
public static void DeleteSingleTypeOrTypeUserFolder(ILibrary library)
{
    //Delete a single type
    LibraryType t1 = library.TypeFolder.Types.Find("type1");
    t1.Delete();

    //Delete a single folder
    LibraryTypeFolder parentFolder = library.TypeFolder;
    LibraryTypeUserFolder f1 = parentFolder.Folders.Find("folder1");
    f1.Delete();
}
```

Modify the following program code to delete a version:

```
public static void DeleteVersion(ILibrary library)
{
    LibraryType singleType = library.TypeFolder.Types.Find("type1");
    LibraryTypeVersion version1 = singleType.Versions.Find(new System.Version(1, 0, 0));
    version1.Delete();
}
```

Modify the following program code to delete a master copy or a user-defined master copy folder:

```
public static void DeleteMasterCopies(ILibrary library)
{
    // Delete master copy
    MasterCopy masterCopy = library.MasterCopyFolder.MasterCopies.Find("myMasterCopy");
    masterCopy.Delete();

    // Delete master copy user folder
    MasterCopyUserFolder masterUserFolder =
library.MasterCopyFolder.Folders.Find("myFolder");
    masterUserFolder.Delete();
}
```

See also

[Accessing master copies \(Page 162\)](#)

7.13 Functions for accessing devices, networks and connections

7.13.1 Open the "Devices & networks" editor

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)

7.13 Functions for accessing devices, networks and connections

Application

You can open the "Devices & networks" editor via the API interface by using one of two methods:

- `ShowHwEditor(View.Topology or View.Network or View.Device)`: Open the "Devices & networks" editor from the project.
- `ShowInEditor(View.Topology or View.Network or View.Device)` : Displays the specified device in the "Devices & networks" editor.

Use the `View` parameter to define the view that is displayed when you open the editor:

- `View.Topology`
- `View.Network`
- `View.Device`

Program code

Modify the following program code to open the "Devices & networks" editor:

```
// Open topology view from project
private static void OpenEditorDevicesAndNetworksFromProject(Project project)
{
    project.ShowHwEditor(Siemens.Engineering.HW.View.Topology);
}
```

Modify the following program code to open the "Devices & networks" editor for a device:

```
// Open topology view for given device
private static void OpenEditorDevicesAndNetworksFromDevice(Device device)
{
    device.ShowInEditor(Siemens.Engineering.HW.View.Topology);
}
```

See also

[Importing configuration data \(Page 753\)](#)

7.13.2 Querying PLC and HMI targets

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)

Application

You can determine whether a software base can be used as PLC target (PlcSoftware) or HMI target in the TIA Portal Openness API.

Program code: PLC target

Modify the following program code to determine if a device item can be used as PLC target:

```
// Returns PlcSoftware
private PlcSoftware GetPlcSoftware(Device device)
{
    DeviceItemComposition deviceItemComposition = device.DeviceItems;
    foreach (DeviceItem deviceItem in deviceItemComposition)
    {
        SoftwareContainer softwareContainer = deviceItem.GetService<SoftwareContainer>();
        if (softwareContainer != null)
        {
            Software softwareBase = softwareContainer.Software;
            PlcSoftware plcSoftware = softwareBase as PlcSoftware;
            return plcSoftware;
        }
    }
    return null;
}
```

Program code: HMI target

Modify the following program code to determine if a device item can be used as HMI target:

```
//Checks whether a device is of type hmitarget
private HmiTarget GetHmiTarget(Device device)
{
    DeviceItemComposition deviceItemComposition = device.DeviceItems;
    foreach (DeviceItem deviceItem in deviceItemComposition)
    {
        SoftwareContainer softwareContainer = deviceItem.GetService<SoftwareContainer>();
        if (softwareContainer != null)
        {
            Software softwareBase = softwareContainer.Software;
            HmiTarget hmiTarget = softwareBase as HmiTarget;
            return hmiTarget;
        }
    }
    return null;
}
```

See also

[Enumerating devices \(Page 228\)](#)

7.13.3 Accessing attributes of an address object

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- For writing access, the PLC is offline.

Application

You can use the TIA Portal Openness API interface to get or set attributes of the address object.

Further you can assign the current process image to an OB.

The following attributes can be accessed:

Attribute name	Data type	Writable	Access	Description
IsochronousMode	BOOL	r/w	Dynamic attribute	Activate/Deactivate isochronousMode
ProcessImage	Int32	r/w	Dynamic attribute	Set/Get process image partition number.
InterruptObNumber	Int64	r/w	Dynamic attribute	Set/Get interrupt organization block number. (classic controller only)
StartAddress	Int32	r/w	Modelled attribute	Set/Get new StartAddress value.

Restrictions

- Attribute `StartAddress`
 - Setting `StartAddress` may implicitly change the `StartAddress` of the opposite IO Type at the name module. Changing of input address changes the output address.
 - Writing access is not supported for all devices.
 - Packed addresses are not supported in TIA Portal Openness
 - Changing an address via TIA Portal Openness will not rewire the assigned tags.
- Attribute `InterruptObNumber`
 - Only accessible in settings with S7-300 or S7-400 controllers. Writing access is supported for S7-400 controllers.

Program code: Get or set attributes of an address object

Modify the following program code to access isochronous mode of an address object:

```
Address address= ...;

// read attribute
bool attributeValue = (bool)address.GetAttribute("IsochronousMode");

// write attribute
address.SetAttribute("IsochronousMode", true);
```

Modify the following program code to access the ProcessImage attribute of an address object:

```
Address address= ...;

// read attribute
int attributeValue = (int)address.GetAttribute("ProcessImage");

// write attribute
address.SetAttribute("ProcessImage", 7);
```

Modify the following program code to access the InterruptObNumber attribute of an address object:

```
Address address= ...;

// read attribute
long attributeValue = (long)address.GetAttribute("InterruptObNumber");

// write attribute
address.SetAttribute("InterruptObNumber", 42L);

// default value = 40
```

Modify the following program code to access the StartAddress attribute of an address object:

```
Address address= ...;

// read attribute
int attributeValue = (int)address.GetAttribute("StartAddress");

// write attribute
address.StartAddress = IntValueStartAddress;
```

Program code: Assign the current process image to an OB

Modify the following program code to assign the current process image to an OB:

```
OB obX =...
Address address= ...;

// assign PIP 5 to obX

address.SetAttribute("ProcessImage", 5);

try
{
    address.AssignProcessImageToOrganizationBlock(obX);
} catch(RecoverableException e) {
    Console.WriteLine(e.Message);
}

// remove this PIP-OB assignment

try
{
    address.AssignProcessImageToOrganizationBlock(null);
} catch(RecoverableException e) {
    Console.WriteLine(e.Message);
}
```

7.13.4 Accessing the channels of a module**Requirement**

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

Signal Modules like analog input modules usually have multiple channels within a single module. Usually, channels provide similar functionality multiple times, e.g. an analog input module with four channels can measure four voltage values at the same time.

To access all channels of a module, the Channels attribute of an device item is used.

Program code: Attributes of channels

Modify the following program code to access attributes of a channel:

```
DeviceItem aiModule = ...
ChannelComposition channels = aiModule.Channels;
foreach (Channel channel in channels)
{
    ... // Work with the channel
}
```

Program code: Identifying attributes

Modify the following program code to get the identifying attribute for each channel:

```
Channel channel = ...
int channelNumber = channel.Number;
ChannelType type = channel.Type;
ChannelIoType ioType = channel.IoType;
```

Program code: Accessing a single channel

Modify the following program code to use the identifying attributes to access a channel directly:

```
DeviceItem aiModule = ...
Channel channel = aiModule.Channels.Find(ChannelType.Analog, ChannelIoType.Input, 0);
... // Work with the channel
```

Channel types

Value	Description
ChannelType.None	The channel type invalid.
ChannelType.Analog	The channel type is analog.
ChannelType.Digital	The channel type is digital.
ChannelType.Technology	The channel type is technology.

Channel IO types

Value	Description
ChannelIoType.None	The channel IO type invalid.
ChannelIoType.Input	An input channel.
ChannelIoType.Output	An output channel.
ChannelIoType.Complex	Complex IO types, e.g. for technological channels.

7.14 Functions on networks

7.14.1 Creating a subnet

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

Subnets can be created in two different ways:

- Create a subnet that is connected to an interface: The type of the interface, where the subnet is created, determines the type of the subnet
- Create a subnet not connected to an interface.

Program code: Create a subnet connected to a node

Modify the following program code to create a subnet:

```
Node node = ...;
Subnet subnet = node.CreateAndConnectToSubnet("NameOfSubnet");
```

The following type identifiers are used:

- System:Subnet.Ethernet
- System:Subnet.Profibus
- System:Subnet.Mpi
- System:Subnet.Asi

Program code: Create a subnet not connected to an interface

Modify the following program code to create a subnet:

```
Project project = ...;
SubnetComposition subnets = project.Subnets;

Subnet newSubnet = subnets.Create("System:Subnet.Ethernet", "NewSubnet");
```

The following type identifiers can be used:

- System:Subnet.Ethernet
- System:Subnet.Profibus
- System:Subnet.Mpi
- System:Subnet.Asi

7.14.2 Accessing subnets

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

For several network related features, e.g. assigning interfaces to a subnet, you have to access subnets in the project. Typically, subnets are aggregated directly on project level.

Program code: Access all subnets of a project

Modify the following program code to access all subnets excluding internal subnets of a project:

```
Project project = ...
foreach (Subnet net in project.Subnets)
{
    ... // Work with the subnet
}
```

Program code: Access a specific subnet

Modify the following program code to access a specific subnet by its name:

```
Project project = ...
Subnet net = project.Subnets.Find("PROFIBUS_1");
{
    ... // Work with the subnet
}
```

Attributes of a subnet

A subnet has the following attributes:

```
Subnet net = ...;
string name = net.Name;
NetType type = net.NetType;
```

Network types

Value	Description
NetType.Unknown	The type of the network is unknown.
NetType.Ethernet	The type of the network is Ethernet.
NetType.Profibus	The type of the network is Profibus.
NetType.Mpi	The type of the network is MPI.
NetType.ProfibusIntegrated	The type of the network is integrated Profibus.
NetType.Asi	The type of the network is ASI.
NetType.PcInternal	The type of the network is PC internal.
NetType.Ptp	The type of the network is PtP.
NetType.Link	The type of the network is Link.
NetType.Wan	The type of the network is Wide Area Network

7.14.3 Accessing internal subnets

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

If a device item is able to compose a subnet, it provides the additional functionality "subnet owner". To access this additional functionality, a specific service of the device item must be used.

Program code: Get the subnet owner role

Modify the following program code to get the subnet owner role:

```
SubnetOwner subnetOwner =
((IEngineeringServiceProvider)deviceItem).GetService<SubnetOwner>();
if (subnetOwner != null)
{
    // work with the role
}
```

Program code: Attributes of a subnet owner

Modify the following program code to access the subnets of a subnet owner:

```
foreach(Subnet subnet in subnetOwner.Subnets)
{
    Subnet internalSubnet = subnet;
}
```

7.14.4 Get type identifier of subnets

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The attribute `TypeIdentifier` is used to identify a subnet. The `TypeIdentifier` is a string consisting of several parts: `<TypeIdentifierType>:<SystemIdentifier>`

Possible values for `TypeIdentifierType` are:

- System

SystemIdentifier

Subnet type	SystemIdentifier
PROFIBUS	Subnet.Profibus
MPI	Subnet.Mpi
Industrial Ethernet	Subnet.Ethernet
ASI	Subnet.Asi
Ptp	Subnet.Ptp

Subnet type	SystemIdentifier
PROFIBUS-Integrated	Subnet.ProfibusIntegrated
PC-Internal	null

Program code

Modify the following program code to get the type identifier for user manageable and separately creatable objects for GSD:

```
Subnet subnet = ...;
string typeIdentifier = subnet.TypeIdentifier;
```

7.14.5 Accessing attributes of a subnet

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

A subnet provides certain mandatory attributes that can be read and/or written. The attributes are only available, if they are available at the UI. Writing is generally only allowed if an attribute can also be changed by the user at the UI. This might vary depending on the type of the subnet. For example, the user can only set the DpCycleTime, if the IsochronousMode is true and the DpCycleMinTimeAutoCalculation is false

Attributes of subnets of type ASI

Attribute	Data type	Writable	Access	Description
Name	string	r/w		Name of the subnet.
NetType	NetType	r		Type of the subnet
SubnetId	string	r	dynam-ic	Unique identification of the subnet. The S7 subnet ID is made up of two numbers separated by a hyphen. One number for the project and one for the subnet. e.g. 4493-1.

Attributes of subnets of type Ethernet

Attribute	Data type	Writa-ble	Access	Description
Name	string	r/w		Name of the subnet.
NetType	NetType	r		Type of the subnet
SubnetId	string	r/w	dynam-ic	Unique identification of the subnet. The S7 subnet ID is made up of two numbers separated by a hyphen. One number for the project and one for the subnet. e.g. 4493-1.
DefaultSubnet	bool	r/w	dynam-ic	true if the subnet is a default subnet. There is at most one default subnet in a project.

Attributes of subnets of type MPI

Attribute	Data type	Writa-ble	Access	Description
Name	string	r/w		Name of the subnet.
NetType	NetType	r		Type of the subnet
SubnetId	string	r/w	dynam-ic	Unique identification of the subnet. The S7 subnet ID is made up of two numbers separated by a hyphen. One number for the project and one for the subnet. e.g. 4493-1.
HighestAddress	int	r/w	dynam-ic	Highest MPI address at subnet.
TransmissionSpeed	BaudRate	r/w	dynam-ic	True if the subnet is a default subnet. There is at most one default subnet in a project.

Attributes of subnets of type PC internal

Attribute	Data type	Writa-ble	Access	Description
Name	string	r		Name of the subnet.
NetType	NetType	r		Type of the subnet
SubnetId	string	r	dynam-ic	Unique identification of the subnet. The S7 subnet ID is made up of two numbers separated by a hyphen. One number for the project and one for the subnet. e.g. 4493-1.

Attributes of subnets of type PROFIBUS

Attribute	Data type	Writa-ble	Access	Description
Name	string	r/w		Name of the subnet.
NetType	NetType	r		Type of the subnet

7.14 Functions on networks

Attribute	Data type	Writa-ble	Access	Description
SubnetId	string	r/w	dynam-ic	Unique identification of the subnet. The S7 subnet ID is made up of two numbers separated by a hyphen. One number for the project and one for the subnet. e.g. 4493-1.
HighestAddress	int	r/w	dynam-ic	Highest PROFIBUS address at subnet.
TransmissionSpeed	BaudRate	r/w	dynam-ic	True if the subnet is a default subnet. There is at most one default subnet in a project.
BusProfile	BusProfile	r/w	dynam-ic	The PROFIBUS profile.
PbCableConfiguration	bool	r/w	dynam-ic	True to enable additional PROFIBUS network settings
PbRepeaterCount	int	r/w	dynam-ic	Number of repeaters for copper cable
PbCopperCableLength	double	r/w	dynam-ic	The length of the copper cable
PbOpticalComponentCount	int	r/w	dynam-ic	Number of OLMs and OBTs of fiber-optical cable.
PbOpticalCableLength	double	r/w	dynam-ic	The length of the fiber-optical cable for the PROFIBUS network in km.
PbOpticalRing	bool	r/w	dynam-ic	True if bus parameter are adapted for an optical ring
PbOlmP12	bool	r/w	dynam-ic	True if OLM/P12 is enabled for bus parameter calculation
PbOlmG12	bool	r/w	dynam-ic	True if OLM/G12 is enabled for bus parameter calculation
PbOlmG12Eec	bool	r/w	dynam-ic	True if OLM/G12-EEC is enabled for bus parameter calculation
PbOlmG121300	bool	r/w	dynam-ic	True if OLM/G12-1300 is enabled for bus parameter calculation
PbAdditionalNetworkDevices	bool	r/w	dynam-ic	True if additional bus devices that don't exist in project will be taken in to account when calculating bus times.
PbAdditionalDpMaster	int	r/w	dynam-ic	Number of unconfigured DP masters.
PbTotalDpMaster	int	r	dynam-ic	Number of total DP masters
PbAdditionalPassiveDevice	int	r/w	dynam-ic	Number of unconfigured DP slaves or passive devices.
PbTotalPassiveDevice	int	r	dynam-ic	Number of total DP slaves or passive devices.
PbAdditionalActiveDevice	int	r/w	dynam-ic	Number of unconfigured active devices with FDL/FMS/S/ communication load.
PbTotalActiveDevice	int	r	dynam-ic	Number of total active devices with FDL/FMS/S/ communication load.
PbAdditionalCommunicationLoad	Communica-tionLoad	r/w	dynam-ic	Rough quantification of the communication load

Attribute	Data type	Writa-ble	Access	Description
PbDirectDateExchange	bool	r/w	dynam-ic	Optimization for direct data exchange.
PbMinimizeTslotForSlaveFailure	bool	r/w	dynam-ic	Minimization for time allocation for slave failure.
PbOptimizeCableConfiguration	bool	r/w	dynam-ic	Optimiazation of the cable configuration.
PbCyclicDistribution	bool	r/w	dynam-ic	True if enable cyclic distribution of bus parameter.
PbTslotInit	int	r/w	dynam-ic	Default value of Tslot.
PbTslot	int	r	dynam-ic	Waiting to receive time (slot time)
PbMinTsdr	int	r/w	dynam-ic	Minimum protocol processing time
PbMaxTsdr	int	r/w	dynam-ic	Maximum protocol processing time
PbTid1	int	r	dynam-ic	Idle time 1
PbTid2	int	r	dynam-ic	Idle time 2
PbTrdy	int	r	dynam-ic	Ready time
PbTset	int	r/w	dynam-ic	Setup time
PbTqui	int	r/w	dynam-ic	Quiet time for modulator
PbTtr	int64	r/w	dynam-ic	The Ttr value in t_Bit
PbTtrTypical	int64	r	dynam-ic	Average response time on bus
PbWatchdog	int64	r/w	dynam-ic	Watchdog
PbGapFactor	int	r/w	dynam-ic	Gab update factor
PbRetryLimit	int	r/w	dynam-ic	Maximum number of retries
IsochronousMode	bool	r/w	dynam-ic	True if constant bus cycle time is enabled.
PbAdditionalPassivDeviceForIsochronousMode	int	r/w	dynam-ic	Number of additional OPs/PGs/TDs etc. that are not configured in this network view.
PbTotalPassivDeviceForIsochronousMode	int	r	dynam-ic	Sum of configured and unconfigured devices, such as OPs/PGs/TDs etc.
DpCycleMinTimeAutoCalculation	bool	r/w	dynam-ic	True if automatic calculation and setting of shortest DP cycle time is enabled.
DpCycleTime	double	r/w	dynam-ic	The DP cycle time.

7.14 Functions on networks

Attribute	Data type	Writa-ble	Access	Description
IsochronousTiToAutoCalculation	bool	r/w	dynam-ic	True if automatic calculation and setting of values of IsochronousTi and IsochronousTo.
IsochronousTi	double	r/w	dynam-ic	Time Ti (read in process values)
IsochronousTo	double	r/w	dynam-ic	Time To (output process values)

Attributes of subnets of type PROFIBUS Integrated

Attribute	Data type	Writa-ble	Access	Description
Name	string	r/w		Name of the subnet.
NetType	NetType	r		Type of the subnet
SubnetId	string	r/w	dynam-ic	Unique identification of the subnet. The S7 subnet ID is made up of two numbers separated by a hyphen. One number for the project and one for the subnet. e.g. 4493-1.
IsochronousMode	bool	r	dynam-ic	Enabled constant bus cycle time.
DpCycleMinTimeAutoCal-culation	bool	r/w	dynam-ic	True if automatic calculation and setting of shortest DP cycle time is enabled.
DpCycleTime	double	r/w	dynam-ic	The DP cycle time.
IsochronousTiToAutoCalcu-lation	bool	r/w	dynam-ic	True if automatic calculation and setting of values of IsochronousTi and IsochronousTo.
IsochronousTi	double	r/w	dynam-ic	Time Ti (read in process values)
IsochronousTo	double	r/w	dynam-ic	Time To (output process values)

Program code

Modify the following program code to get or set the attributes of a subnet:

```

Subnet subnet = ...;

string nameValue = subnet.Name;
NetType nodeType = (NetType)subnet.NetType;
string subnetId = ((IEngineeringObject)subnet).GetAttribute("SubnetId");

subnet.Name = "NewName";
subnet.SetAttribute("Name", "NewName");

bool isDefaultSubnet = ((IEngineeringObject)subnet).GetAttribute("DefaultSubnet");

```

Baud rates

Value	Description
BaudRate.None	The baud rate is unknown.
BaudRate.Baud9600	9.6 kBaud
BaudRate.Baud19200	19.2 kBaud
BaudRate.Baud45450	45.45 kBaud
BaudRate.Baud93700	93.75 kBaud
BaudRate.Baud187500	187.5 kBaud
BaudRate.Baud500000	500 kBaud
BaudRate.Baud1500000	1.5 MBaud
BaudRate.Baud3000000	3 MBaud
BaudRate.Baud6000000	6 MBaud
BaudRate.Baud12000000	12 MBaud

Bus profiles

Value	Description
BusProfile.None	The bus profile is unknown.
BusProfile.DP	The type of the network is DP.
BusProfile.Standard	The type of the network is Standard.
BusProfile.Universal	The type of the network is Universal.
BusProfile.UserDefined	The type of the network is user defined.

Communication load

Value	Description
CommunicationLoad.None	No valid communication load.
CommunicationLoad.Low	Typically used for DP, no great data communication apart from DP.
CommunicationLoad.Medium	Typically used for mixed operations featuring DP and other communication services, such as for S7 communication, when DP has strict time requirements and for average acyclic volumes of communication.
CommunicationLoad.High	For mixed operations featuring DP and other communication services, such as for S7 communication, when DP has loose time requirements and for high acyclic volumes of communication.

7.14.6 Deleting a global subnet

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Program code

Modify the following program code to delete a global subnet within a project.:

```
Project project = ...;
SubnetComposition subnets = projects.Subnets;

// delete subnet
Subnet subnetToDelete = ...;
subnetToDelete.Delete();
```

7.14.7 Enumerate all participants of a subnet

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

Enumeration of all participants on a subnet.

Program code

Modify the following program code to enumerate dp master systems from subnet:

```
Subnet subnet = ...;
foreach (Node node in subnet.Nodes)
{
    // work with the node
}
```

7.14.8 Enumerate IO systems of a subnet

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

Enumeration of IoSystem provides all io systems that are present on a subnet. The class IoSystem represents the master systems and the io systems.

Program code

Modify the following program code to enumerate dp master systems from subnet:

```
Subnet subnet = ...;
foreach (IoSystem ioSystem in subnet.IoSystems)
{
    // work with the io system
}
```

7.14.9 Accessing nodes

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The role interface aggregates nodes to access attributes that are related to the address and subnet assignment of an interface.

The name of a node can be seen in the attributes of an interface in TIA Portal . The NodId is a unique identifier for every node aggregated at an interface, its value can only be seen via TIA Portal Openness.

Program code

Modify the following program code to access all nodes of an interface:

```
NetworkInterface itf = ...
foreach (Node node in itf.Nodes)
{
    ... // Work with the node
}
```

Most interfaces provide only a single node, therefore, usually the first node is used:

```
NetworkInterface itf = ...
Node node = itf.Nodes.First();
{
    ... // Work with the node
}
```

Nodes provide their names and types and Nodelds as attributes:

```
Node node = ...
string name = node.Name;
NetType type = node.NodeType;
string id = node.NodeId;
```

7.14.10 Accessing attributes of a node

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

A device item provides certain mandatory attributes that can be read and/or written. The attributes are only available if they are available at the UI. Writing is generally only allowed if an attribute can also be changed by the user at the UI. This might vary depending on the type of the device item. For example, the user can only set the RouterAddress if the RouterUsed is true. If the user changes the SubnetMask at IO controller, Subentmask on all IO devices will be also changed to the same value.

Attributes of a node of type ASI

Attributes	Data type	Writa-ble	Access	Description
Name	string	r		Name of the node.
NodeId	string	r		ID of the node.
NodeType	NetType	r		A node gets his type from the subnet.
Address	string	r/w	dynam-ic	Additional attribute for AS-i slaves.

Attributes of a node of type Ethernet

Attributes	Data type	Writa-ble	Access	Description
Name	string	r		Name of the node.
NodeId	string	r		ID of the node.
NodeType	NetType	r/w some-times r/o		A node gets his type from the subnet.
UseIsoProtocol	bool	r/w	dynam-ic	True, if ISO protocol should be used
MacAddress	string	r/w	dynam-ic	e.g. 01-80-C2-00-00-00
UseIpProtocol	bool	r/w	dynam-ic	This value can be read even if it is not visible at the corresponding TIA UI control.
IpProtocolSelection	enum	r/w	dynam-ic	
Address	string	r/w	dynam-ic	only IPv4 and no IPv6 is supported
SubnetMask	string	r/w	dynam-ic	
UseRouter	bool	r/w	dynam-ic	
RouterAddress	string	r/w	dynam-ic	
DhcpClientId	string	r/w	dynam-ic	
PnDeviceNameSetDirectly	bool	r/w	dynam-ic	PROFINET device name is set directly at the device. Not available for every device.
PnDeviceNameAutoGeneration	bool	r/w	dynam-ic	PROFINET device name is created automatically.
PnDeviceName	string	r/w	dynam-ic	Unique device name in subnet.
PnDeviceNameConverted	string	r	dynam-ic	Device name converted for system internal usage.

Attributs of a node of type MPI

Attribut	Data type	Writable	Access	Description
Name	string	r		Name of the node.
NodeId	string	r		ID of the node.
NodeType	NetType	r		A node gets his type from the subnet.
Address	string	r/w	dynam-ic	

Attributs of a node of type PC internal

Attribut	Data type	Writable	Access	Description
Name	string	r		Name of the node.
NodeId	string	r		ID of the node.
NodeType	NetType	r		A node gets his type from the subnet.

Attributs of a node of type PROFIBUS

Attribut	Data type	Writable	Access	Description
Name	string	r		Name of the node.
NodeId	string	r		ID of the node.
NodeType	NetType	r		A node gets his type from the subnet.
Address	string	r/w	dynam-ic	The network address of the node. The type of address depends on the node type (e.g. IP address for PROFINET nodes, PROFIBUS address for PROFIBUS nodes)

Attributs of a node of type PROFIBUS integrated

Attribut	Data type	Writable	Access	Description
Name	string	r		Name of the node.
NodeId	string	r		ID of the node.
NodeType	NetType	r		A node gets his type from the subnet.
Address	string	r	dynam-ic	

Program code: Attributes of a node

Modify the following program code to get or set the attributes of a node:

```
Node node = ...;
string nameValue = node.Name;
NetType nodeType = (NetType)node.NodeType;
node.NodeType = NetType.Mpi;
```

Program code: Dynamic attributes

Modify the following program code to get or set dynamic node attributes:

```
Node node = ...;
var attributeNames = new []
{
    "Address", "SubnetMask", "RouterAddress", "UseRouter", "DhcpClientId",
    "IpProtocolSelection"
};
foreach (var attributeName in attributeNames)
{
    object attributeValue = ((IEngineeringObject)node).GetAttribute(attributeName);
}
```

Protocol selection

Value	Description
IpProtocolSelection.None	Error value
IpProtocolSelection.Project	IP suite configured within project.
IpProtocolSelection.Dhcp	IP suite managed via DHCP protocol. DHCP Client ID necessary.
IpProtocolSelection.UserProgram	IP suite set via FB (function block).
IpProtocolSelection.OtherPath	IP suite set via other methods, for example PST tool.
IpProtocolSelection.VialoController	IP suite set via IO Controller in runtime.

Net type

Value	Description
NetType.Asi	Net type is ASI.
NetType.Ethernet	Net type is Ethernet.
NetType.Link	Net type is Link.
NetType.Mpi	Net type is MPI.
NetType.PcInternal	Net type is PC internal.
NetType.Profibus	Net type is PROFIBUS.
NetType.ProfibusIntegrated	Net type is PROFIBUS integrated.
NetType.Ptp	Net type is PTP.

7.14 Functions on networks

Value	Description
NetType.Wan	Net type is Wide Area Network (WAN).
NetType.Unknown	Net type is Unknown.

7.14.11 Connecting a node to a subnet**Requirement**

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Program code

Modify the following program code to assign a node (device, interface) to a network:

```
Node node = ...;
Subnet subnet = ...;
node.ConnectToSubnet(subnet);
```

7.14.12 Disconnect a node from a subnet**Requirement**

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Program code

Modify the following program code to disconnect a node (device, interface) from a network:

```
Node node = ...;
node.DisconnectFromSubnet();
```

7.14.13 Creating an IO system

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

An io system is created by calling the action `IoController.CreateIoSystem("name")` on an object of the type `IoController`. In case name is null or `String.Empty` the default name will be used. The `io controller` is aquired by accessing the attribute `IoControllers` object on the `NetworkInterface`. The `IoControllers` navigator returns one `IoController` object.

Prerequisites for creating an io system:

- The interface of the io controller is connected to a subnet.
- The io controller has no io system.

Program code

Modify the following program code to create an io system:

```
using System.Linq;
...
NetworkInterface interface = ...;
IoSystem ioSystem = null;

// Interface is configured as io controller
if((interface.InterfaceOperatingMode & InterfaceOperatingModes.IoController) != 0)
{
    IoControllerComposition ioControllers = interface.IoControllers;
    IoController ioController = ioControllers.First();
    if(ioController != null)
    {
        ioSystem = ioController.CreateIoSystem("io system");
    }
}
```

7.14.14 Accessing the attributes of an IO system

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The master system and the io system will both be represented by the class IoSystem.

Program code: Attributes of an io system

Modify the following program code to get the attributes of the IoSystem:

```
NetworkInterface itf = ...  
foreach (IIoController ioController in itf.IoControllers)  
{  
    IoSystem ioSystem = ioController.IoSystem;  
    int ioSystemNumber = ioSystem.Number;  
    string ioSystemName = ioSystem.Name;  
}
```

Program code: Subnet of an io system

Modify the following program code to navigate to the subnet the io system is assigned to:

```
Subnet subnet = ioSystem.Subnet;
```

7.14.15 Connecting an IO connector to an IO system

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

Use the action ConnectToIoSystem(IoSystem ioSystem) of IoConnector to connect a profinet or a DP IoConnector to an existing io system.

Use the action GetIoController to navigate to the remote IoController. For further information how to navigate to the local IoConnector and the io system see Get master system or IO system of an interface (Page 203).

Prerequisites:

- The IoConnector is not yet connected to an io system.
- The IoConnector interface is connected to the same subnet as the interface of the desired IoController.

Program code

Modify the following program code:

```
IoSystem ioSystem = ...;
IoConnector ioConnector = ...;
ioConnector.ConnectToIoSystem();
IoController ioController = ioConnector.GetIoController();
```

7.14.16 Get master system or IO system of an interface

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The service NetworkInterface provides the navigator IoControllers, each IoController in turn provides the navigator IoSystem. The class IoSystem represents the master systems and the io systems. The io device and the slave are both named io device.

- The IoControllers navigator returns IoController objects, if the network interface can have an io system. At the moment only one io controller will be returned.
- The IoConnectors navigator returns IoConnector objects, if the network interface can be connected to an io system as an io device. At the moment only one io connector will be returned.

Program code: Get the io system of the IoController

Modify the following program code to get the io system of the IoController:

```
NetworkInterface itf = ...
foreach (IoController ioController in itf.IoControllers)
{
    IoSystem ioSystem = ioController.IoSystem;
    // work with the io system
}
```

Program code: Get the io system of the IoConnector

Modify the following program code to get the io system of the IoConnector:

```
NetInterface itf = ...
foreach (IoConnector ioConnector in itf.IoConnectors)
{
    IoSystem ioSystem = ioConnector.ConnectedIoSystem;
    // work with the io system
}
```

7.14.17 Get an IO Controller

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Program code

Currently only configurations with one IoController are possible. An IoController does not provide any modelled attributes or actions.

Program code

Modify the following program code to get the io controller:

```
NetworkInterface itf = ...
foreach (IoController ioController in itf.IoControllers)
{
    // work with the io controller
}
```

7.14.18 Get an IO Connector

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

An IoConnector does provide the modelled attributes or actions.

The following attribute and actions are available at the IoConnector:

Actions

Action	Signature	Description
ConnectToIoSystem	void ConnectToIoSystem(Siemens.Engineering.HW.IoSystem ioSystem)	To connect an IoConnector to an existing DP master system
DisconnectFromIoSystem	void DisconnectFromIoSystem()	To disconnect an IoConnector from an existing io system
GetIoController	Siemens.Engineering.HW.IoController GetIoController()	Returns the IO controller for this connector

Links

Link	Type	Access
ConnectedToIoSystem	Siemens.Engineering.HW.IoSystem	read-only

Program code

Modify the following program code to get the io connector:

```
NetworkInterface itf = ...
foreach (IoConnector ioConnector in itf.IoConnectors)
{
    // work with the IoConnector
}
```

7.14.19 Disconnecting an IO connector from an IO system or a DP mastersystem

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

Use the action `DisconnectFromIoSystem()` of `IoConnector` to disconnect an `IoConnector` from an existing io system or an existing DP mastersystem.

For further information how to navigate to the local `IoConnector` and the io system see Get master system or IO system of an interface (Page 203).

Program code

Modify the following program code:

```
IoSystem ioSystem = ...;  
IoConnector ioConnector = ...;  
  
ioConnector.DisconnectFromIoSystem();
```

7.14.20 Accessing attributes of a DP mastersystem

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

A DP Mastersystem provides certain attributes that can be read and/or written. The attributes are only available if they are available at the UI. Writing is generally only allowed if an attribute can also be changed by the user at the UI. This might vary depending on the DP Master and the DP Slaves which are assigned to this DP Mastersystem.

Attributes of a dp mastersystem

Attribute	Data type	Writable	Access	Description
Name	string	r/w		
Number	int	r/w		The property Number accepts values that cannot be set via UI. In this case the compile will fail.

Program code: Get attributes

Modify the following program code to get attributes:

```
IoSystem dpMastersystem = ...;

string name = dpMastersystem.Name;
int number = dpMastersystem.Number;
```

Program code: Set attributes

Modify the following program code to set attributes:

```
IoSystem dpMastersystem = ...;

dpMastersystem.Name ="myDpMastersystem"
dpMastersystem.Number=42;
```

7.14.21 Accessing attributes of a profinet io system

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

An IO System provides certain attributes that can be read and/or written. The attributes are only available if they are available at the UI. Writing is generally only allowed if an attribute can also be changed by the user at the UI. This might vary depending on the IO Controller and the IO Devices which are assigned to this IO System.

Attributes of a PROFINET io System

Attribute	Data type	Writable	Access	Description
MultipleUseIoSystem	bool	r/w	dynam-ic	
Name	string	r/w		
Number	int	r/w		The attribute Number accepts values that cannot be set via UI. In this case the compile will fail.
UseIoSystemNameAsDeviceNameExtension	bool	r/w	dynam-ic	If MultipleUseIoSystem is set to TRUE, UseIoSystemNameAsDeviceNameExtension will be set to FALSE and write access is not possible.
MaxNumberIWlanLinksPerSegment	int	r/w	dynam-ic	

Program code: Get attributes

Modify the following program code to get attributes

```
IoSystem ioSystem = ...;
string name = ioSystem.Name;
```

Program code: Set attributes

Modify the following program code to set attributes:

```
IoSystem ioSystem = ...;
ioSystem.Name = "IOSystem_1";
```

Program code: Get attributes with dynamic access

Modify the following program code to get the values of dynamic attributes:

```
IoSystem ioSystem = ...;
var attributeNames = new[]
{
    "MultipleUseIoSystem", "UseIoSystemNameAsDeviceNameExtension",
    "MaxNumberIWlanLinksPerSegment"
};
foreach (var attributeName in attributeNames)
{
    object attributeValue = ((IEngineeringObject)ioSystem).GetAttribute(attributeName);
```

Program code: Set attributes with dynamic access

Modify the following program code to set the values of dynamic attributes:

```
IoSystem ioSystem = ...;  
((IEngineeringObject)ioSystem).SetAttribute("MultipleUseIoSystem", true);
```

7.14.22 Deleting a DP master system

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Program code: Deleting a PROFINET io system

Modify the following program code to delete a PROFINET io system:

```
IoController ioController = ...;  
IoSystem ioSystem = ioController.IoSystem;  
  
ioSystem.Delete();
```

7.14.23 Deleting a profinet io system

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Program code

Modify the following program code to delete a profinet io system:

```
IoController ioController = ...;  
IoSystem ioSystem = ioController.IoSystem;  
ioSystem.Delete();
```

7.14.24 Creating a DP master system

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

A DP master system is created by calling the action `CreateIoSystem(string nameOfIoSystem)` on an object of the type `IoController`. The `io controller` is acquired by accessing the attribute `IoControllers` object on the `NetworkInterface`.

Prerequisites for creating a DP master system:

- The interface of the `io controller` is connected to a subnet.
- The `io controller` has no `io system`.

Program code

Modify the following program code to create a dp master system:

```
using System.Linq;
...
NetworkInterface interface = ...;
IoSystem dpMasterSystem = null;

// Interface is configured as master or as master and slave
if((interface.InterfaceOperatingMode & InterfaceOperatingModes.IoController) != 0)
{
    IoControllerComposition ioControllers = interface.IoControllers;
    IoController ioController = ioControllers.First();
    if(ioController != null)
    {
        dpMasterSystem = ioController.CreateIoSystem("dp master system");
    }
}
```

7.14.25 Accessing port interconnection information of port device item

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

NetworkPort provides the link ConnectedPorts which is an enumeration of ports to access all interconnected partner ports of a port.

It is only possible to interconnect ports which can also be interconnected in the TIA UI, e.g. it is not possible to interconnect two ports of the same Ethernet interface. Recoverable exception are thrown

- if there is already an interconnection to the same partner port
- when trying to interconnect two ports which cannot be interconnected
- when trying to create a second interconnection to a port which does not support alternative partners

Program code: Get the port interconnection

Modify the following program code to get the port interconnection information of a port device item:

```
NetworkPort port = ...;
foreach (NetworkPort partnerPort in port.ConnectedPorts)
{
    // work with the partner port
}
```

Program code: Create port interconnections

Modify the following program code:

```
NetworkPort port1 = ...;
NetworkPort port2 = ...;
port1.ConnectToPort(port2);

// port supports alternative partners
NetworkPort port1 = ...;
NetworkPort port2 = ...;
NetworkPort port3 = ...;
port1.ConnectToPort(port2);
port1.ConnectToPort(port3);
```

Program code: Delete port interconnection

Modify the following program code:

```
NetworkPort port1 = ...;
NetworkPort port2 = ...;
port1.DisconnectFromPort(port2);
```

7.14.26 Attributes of port inter-connection

Attributes for port inter-connections

TIA Portal Openness supports the following attributes for port inter-connections. If the attributes are available in UI, the corresponding attributes can also be accessed through TIA Portal Openness. If the user has access to modify the attributes in UI, then they can also be modified via TIA Portal Openness.

Attribute name	Data type	Writable	Access	Description
MediumAttachment-Type	MediumAttachment-Type	Read-only	Dynamic attribute	
CableName	CableName	Read-Write	Dynamic attribute	
AlternativePartner-Ports	Boolean	Read-Write	Dynamic attribute	Only available if tool-changer functionality is supported.
SignalDelaySelection	SignalDelaySelection	Read-Write	Dynamic attribute	
CableLength	CableLength	Read-Write	Dynamic attribute	
SignalDelayTime	Double	Read-Write	Dynamic attribute	

Enum values of port inter-connection attributes

The Enum MediumAttachmentType has following values.

Value	Description
MediumAttachmentType.None	Attachment type cannot be determined.
MediumAttachmentType.Copper	Attachment type is copper.
MediumAttachmentType.FiberOptic	Attachment type is fiber optic.

The Enum CableName has following value

Value	Description
CableName.None	No cable name is specified
CableName.FO_Standard_Cable_9	FO standard cable GP (9 µm)
CableName.Flexible_FO_Cable_9	Flexible FO cable (9 µm)
CableName.FO_Standard_Cable_GP_50	FO standard cable GP (50 µm)
CableName.FO_Trailing_Cable_GP	FO trailing cable / GP
CableName.FO_Ground_Cable	FO ground cable
CableName.FO_Standard_Cable_62_5	FO standard cable (62.5 µm)
CableName.Flexible_FO_Cable_62_5	Flexible FO cable (62.5 µm)
CableName.POF_Standard_Cable_GP	POF standard cable GP
CableName.POF_Trailing_Cable	POF trailing cable
CableName.PCF_Standard_Cable_GP	PCF trailing cable / GP
CableName.GI_POF_Standard_Cable	GI-POF standard cable
CableName.GI_POF_Trailing_Cable	GI-POF trailing cable
CableName.GI_PCF_Standard_Cable	GI-PCF standard cable
CableName.GI_PCF_Trailing_Cable	GI-PCF trailing cable
CableName.GI_POF_Standard_Cable	GI-POF standard cable
CableName.GI_POF_Trailing_Cable	GI-POF trailing cable
CableName.GI_PCF_Standard_Cable	GI-PCF standard cable
CableName.GI_PCF_Trailing_Cable	GI-PCF trailing cable

The Enum SignalDelaySelection has following values.

Value	Description
SignalDelaySelection.None	
SignalDelaySelection.CableLength	CableLength is used to define the signal delay.
SignalDelaySelection.SignalDelayTime	SignalDelayTime is used to define the signal delay

The Enum CableLength has following values

Value	Description
CableLength.None	CableLength is not specified.
CableLength.Length20m	Cable length is 20m.
CableLength.Length50m	Cable length is 50m.
CableLength.Length100m	Cable length is 100m.
CableLength.Length1000m	Cable length is 1000m.
CableLength.Length3000m	Cable length is 3000m.

Attributes of port options

The attributes of port options are given below.

Attribute name	Data type	Writable	Access
PortActivation	bool	Read-Write	Dynamic attribute
TransmissionRateAnd-Duplex	TransmissionRateAnd-Duplex	Read-Write	Dynamic attribute
PortMonitoring	bool	Read-Write	Dynamic attribute
TransmissionRateAuto-Negotiation	bool	Read-Write	Dynamic attribute
EndOfDetectionOfAc-cessibleDevices	bool	Read-Write	Dynamic attribute
EndOfTopologyDiscov-ery	bool	Read-Write	Dynamic attribute
EndOfSyncDomain	bool	Read-Write	Dynamic attribute

The Enum TransmissionRateAndDuplex has following values.

Value	Description
TransmissionRateAndDuplex.None	
TransmissionRateAndDuplex.Automatic	Automatic
TransmissionRateAndDuplex.AUI10Mbps	10 Mbps AUI
TransmissionRateAndDuplex.TP10MbpsHalfDu-plex	TP 10 Mbps half duplex
TransmissionRateAndDuplex.TP10MbpsFullDu-plex	TP 10 Mbps full duplex
TransmissionRateAndDuplex.AsyncFib-er10MbpsHalfDuplex	async fiber 10Mbit/s half duplex mode
TransmissionRateAndDuplex.AsyncFib-er10MbpsFullDuplex	async fiber 10Mbit/s full duplex mode
TransmissionRateAndDuplex.TP100MbpsHalfDu-plex	TP 100 Mbps half duplex
TransmissionRateAndDuplex.TP100MbpsFullDu-plex	TP 100 Mbps full duplex
TransmissionRateAndDuplex.FO100MbpsFullDu-plex	FO 100 Mbps full duplex
TransmissionRateAndDuplex.X1000MbpsFullDu-plex	X1000 Mbps full Duplex
TransmissionRateAndDuplex.FO1000MbpsFull-DuplexLD	FO 1000 Mbps full duplex LD
TransmissionRateAndDuplex.FO1000MbpsFull-Duplex	FO 1000 Mbps full Duplex
TransmissionRateAndDuplex.TP1000MbpsFull-Duplex	TP 1000 Mbps full duplex

Value	Description
TransmissionRateAndDuplex.FO10000MbpsFull-Duplex	FO 10000 Mbps full Duplex
TransmissionRateAndDuplex.FO100MbpsFullDuplexLD	FO 100 Mbps full duplex LD
TransmissionRateAndDuplex.POFPCF100MbpsFullDuplexLD	POF/PCF 100 Mbps full duplex

7.14.27 Accessing the attributes of a port

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

If a device item is a port, it provides additional functionality over a simple device item.

- It is possible to access the linked partner ports of the port
- It is possible to access the interface of the port

To access this additional functionality, the NetworkPort feature, a specific service of the device item, must be used.

Program code: Accessing a port

Modify the following program code to access attributes of a channel:

```
NetworkPort port = ((IEngineeringServiceProvider)deviceItem).GetService<NetworkPort>();
if (port != null)
{
    ... // Work with the port
}
```

Attributes of a port

A port has the following attributes:

```
NetworkPort port = ....;
var connectedPorts = port.ConnectedPorts;
var myInterface = port.Interface;
```

7.14.28 Enumerate DP master systems of a subnet

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

Enumeration of IoSystem provides all DP mastersystems that are present on a subnet. The class IoSystem represents the master systems and the io systems.

Program code

Modify the following program code to enumerate DP master systems from subnet:

```
Subnet subnet = ...;
foreach (IoSystem ioSystem in subnet.IoSystems)
{
    // work with the io system
}
```

7.14.29 Enumerate assigned IO connectors

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The class IoSystem represents the master systems and the io systems.

It is used for:

- Enumeration of io connectors of a dp mastersystem
- Enumeration of io connectors of a profinet io system

Program code

Modify the following program code to enumerate assigned io connectors of the dp mastersystem:

```
IoSystem ioSystem = ...;
foreach (IoConnector ioConnector in ioSystem.ConnectedIoDevices)
{
    // work with the io connector
}
```

7.14.30 Connecting a DP IO connector to a DP master system

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

Use the action ConnectToIoSystem(IoSystem ioSystem) of IoConnector to connect an IoConnector to an existing DP mastersystem.

Use the action GetIoController to navigate to the remote IoController. For further information how to navigate to the local IoConnector and the io system see Get master system or IO system of an interface (Page 203).

Prerequisites:

- The IoConnector is not yet connected to an io system.
- The IoConnector interface is connected to the same subnet as the interface of the desired IoController.

Program code

Modify the following program code:

```
IoSystem ioSystem = ...;
IoConnector ioConnector = ...;
ioConnector.ConnectToIoSystem(ioSystem);
IoController ioController = ioConnector.GetIoController();
```

7.14.31 Accessing AS-i profile and parameter attributes for virtual slaves

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)

Application

The TIA Portal Openness supports the following additional AS-i profile parameters for the Virtual Slaves of the CTT5 AS-i Slave using the StructuredData names:

Name	Description
AsiProfileVirtualSlave1	This contains the AS-i profile parameters for the Virtual Slave 1
AsiProfileVirtualSlave2	This contains the AS-i profile parameters for the Virtual Slave2
AsiProfileVirtualSlave3	This contains the AS-i profile parameters for the Virtual Slave 3

The AS-i slave parameters for the Virtual slaves can be found below:

Name	Description
AsiParamVirtualSlave1	This contains the AS-i parameter for the Virtual Slave 1
AsiParamVirtualSlave2	This contains the AS-i parameter for the Virtual Slave 2
AsiParamVirtualSlave3	This contains the AS-i parameter for the Virtual Slave 3

Program code

Modify the following program code to get and set the additional attributes of AS-i Slaves:

```
DeviceItem slaveModule= ...;
var structuredDataNamesProfile = new []
{
    "AsiProfileVirtualSlave1", "AsiProfileVirtualSlave2", "AsiProfileVirtualSlave3"
};
var structuredDataNamesAsiParam = new []
{
    "AsiParamVirtualSlave1", "AsiParamVirtualSlave2", "AsiParamVirtualSlave3"
};
var attributeNamesProfile = new []{ "AsiProfileID", "AsiProfileIO", "AsiProfileID2",
    "AsiProfileID1" };
string attributeNameAsiParam = "AsiSlaveParameter"
foreach (var structuredDataName in structuredDataNamesProfile)
{
    foreach (var attributeName in attributeNamesProfile)
    {
        StructuredData structuredData =
        (StructuredData)slaveModule.GetAttribute(structuredDataName);
        //get
        UInt32 attributeValue = (UInt32)structuredData.GetAttribute(attributeName);
        //set
        slaveHead.SetAttribute(attributeName, (UInt32)5);
    }
}
foreach (var structuredDataName in structuredDataNamesAsiParam)
{
    StructuredData structuredData =
    (StructuredData)slaveModule.GetAttribute(structuredDataName);
    //get
    UInt32 attributeValue = (UInt32)structuredData.GetAttribute(attributeNameAsiParam);
    //set
    slaveHead.SetAttribute(attributeName, (UInt32)5);
}
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.15 Functions on devices

7.15.1 Mandatory attributes of devices

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

Every device or device item provides certain mandatory attributes which can be read and/or written. These attributes are always the same as in the TIA Portal user interface.

The following attributes are supported in Openness:

Attribute name	Data type	Writable	Access	Comment
Author	string	read/write	dynamic	
Comment	string	read/write	dynamic	sometimes only read access
CommentML	MultilingualTextItem	read/write	dynamic	sometimes only read access
IsGsd	bool	read		TRUE, if the device description is installed via GSD/GSDML
Name	string	read/write		sometimes only read access
TypeIdentifier	string	read		
TypeName	string	read	dynamic	

Program code: Mandatory attributes of a device

Modify the following program code to get the mandatory attributes of a device:

```
Device device = ...;
string nameValue = device.Name;
bool isGsdValue = device.IsGsd;
```

Program code: Mandatory attributes with dynamic access

Modify the following program code to get the attributes item with dynamic access:

```
Device device = ...;
var attributeNames = new[] {
    "TypeName", "Author", "Comment"
};
foreach (var attributeName in attributeNames) {
    object attributeValue = ((IEngineeringObject)device).GetAttribute(attributeName);
}
```

7.15.2 Get type identifier of devices and device items

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The attribute `TypeIdentifier` is used to identify a hardware object that is creatable via TIA Portal Openness API. The `TypeIdentifier` is a string consisting of several parts: `<TypeIdentifierType>:<Identifier>`

Possible values for `TypeIdentifierType` are:

- OrderNumber
- GSD
- System

OrderNumber

OrderNumber is the common **TypeIdentifier** for all modules present in the hardware catalog.

Format of type identifier	Example	Specifics
<OrderNumber>	OrderNumber:3RK1 200-0CE00-0AA2	
<OrderNumber>/<FirmwareVersion>	OrderNumber:6ES7 510-1DJ01-0AB0/V2.0	Firmware version is optional in case it does not exist or there is only one versionexisting in the system. Be care
<OrderNumber>//<AdditionalTypeIdentifier>	OrderNumber:6AV2 124-2DC01-0AX0//Landscape	The additional type identifier might be necessary in case that OrderNumber and FirmwareVersion do not lead to a unique match in the system.

Note

There are a few modules in the hardware catalog which use "wildcard" characters in the order number to represent a certain cluster of real hardware, e.g. the different lengths of S7-300 racks. In this case the specific **OrderNumber** and the "wildcard"-**OrderNumber** can both be used to create an instance of the hardware object. However you cannot generically use wildcards at any position.

GSD

This is the identifier used for modules that are added to the TIA Portal via GSD or GSDML.

Format of type identifier	Example	Specifics
<GsdName>/<GsdType>	GSD:SIEM8139.GSD/DAP	GsdName is the name of the GSD or GSDML in uppercase letters.
<GsdName>/<GsdType>/<GsdId>	GSD:SIEM8139.GSD/M/4	<p>GsdType is one of the following:</p> <ul style="list-style-type: none"> • D: Device • R: Rack • DAP: HeadModule • M: Module • SM: Submodule <p>GsdId is a identifier for the type.</p>

System

This is the identifier for objects, which cannot be determined using OrderNumber or GSD.

Format of type identifier	Example	Specifics
<SystemTypeIdentifier>	System:Device.S7300	SystemTypeIdentifier is the primary identifier of an object.
<SystemTypeIdentifier>/<AdditionalTypeIdentifier>	GSD:SIEM8139.GSD/M/4	AdditionalTypeIdentifier might be necessary in case the SystemTypeIdentifier is not unique. The prefix for certain object types are: <ul style="list-style-type: none"> • Connection. • Subnet. • Device. • Rack.

Program code

Modify the following program code to get the type identifier for user manageable and separately creatable objects for GSD:

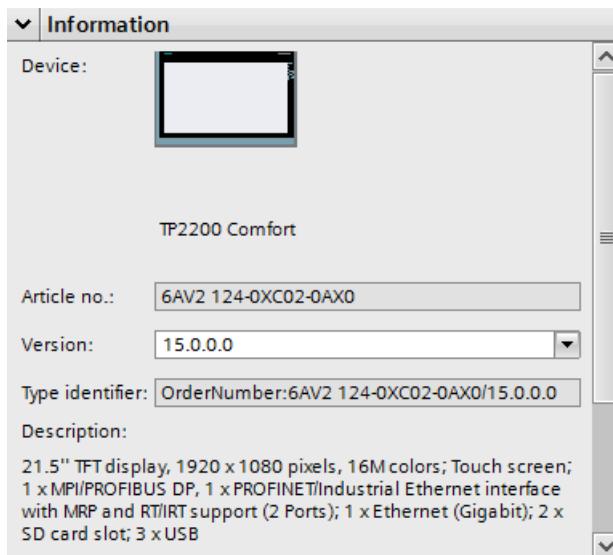
```
HardwareObject hardwareObject = ....;
string typeIdentifier = hardwareObject.TypeIdentifier;
```

Displaying type identifiers in TIA Portal

If you need to know a type identifier you inquire it in TIA Portal as follows:

1. Enable the setting "Enable display of the type identifier for devices and modules" in "Options > Settings > Hardware configuration > Display of the type identifier".
2. Open the editor "Devices & networks".
3. Select a device in the Catalog.

The type identifier is displayed in the viewlet "Information"



7.15.3 Set App ID on device and device items

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal
- A project is open
See Opening a Project

Introduction

You can use the TIA Portal Openness to set the Application ID on device and device items of TIA Portal project, so that these Application IDs are retained in the current session of TIA Portal for later use.

You can use the TIA Portal Openness to set the Application ID by providing "Application Key" and "Application Value" pair to the API.

The following constraints applicable for the Application ID of device and device item object:

- An object of type supporting this feature can have maximum of 64 Application IDs available for it
- The Application IDs Key or/and Value length can be maximum 128 Characters in length.
- The Application ID Key for a given object is unique
- Only one Application ID can be set using one "Application Key"
- When different Application Value is set with the same "Application Key", the last set Application Value is retained

Program code

Modify the following program code to set the Application ID for device and device item object:

```
Device device = ...;
// Ask for Service CustomIdentityProvider on the device/device item
var customIdentityProviderService = device.GetService<CustomIdentityProvider>();
//Set the Application ID (Key-Value) pair
customIdentityProviderService.Set("Application_Key", "Application_Value");
```

7.15.4 Get App ID on device and device Items

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a Project (Page 106)

Introduction

You can use the TIA Portal Openness to get the Application ID on device and device items of TIA Portal project, if the device or device items is already set with an Application ID.

Application IDs are stored as "Application Key" and "Application Value" pair as part of TIA Portal. You can use the TIA Portal Openness to get the Application ID value by providing "Application Key".

Program code

Modify the following program code to get the Application ID value for an object, which was already set using the Application Key:

```
Device device = ...;
// Ask for Service CustomIdentityProvider on the device/device item
var customIdentityProviderService = device.GetService<CustomIdentityProvider>();
customIdentityProviderService.Set("Application_Key", "Application_Value");
//Get the Application ID (Value) for the given Application Key
var applicationValue = customIdentityProviderService.Get("Application_Key");
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.15.5 Remove App ID on device and device items

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open
See [Opening a Project \(Page 106\)](#)

Principle

You can use the TIA Portal Openness to remove Application ID (Custom Identity) for a device and device items of TIA Portal project, so that device and device item objects are updated with relevant Application ID.

A CustomIdentityNotFoundException will be thrown, if the Application ID which is passed as an argument does not exist.

Program code

Modify the following program code to remove the Application ID for device and device items:

```
Device device = ...;
// Ask for Service CustomIdentityProvider on the device/device item
var customIdentityProviderService = device.GetService<CustomIdentityProvider>();
//Remove the CustomIdentity corresponding to Application ID
try
{
customIdentityProviderService.Remove("Application_Key");
}
catch(CustomIdentityNotFoundException ex)
{
// When CustomIdentity is not found for a the given key "Application_Key".
}
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.15.6 Creating a device

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)

Application

A device can be created via two methods, within a project or a device group:

- Create a device via a device item type identifier like in TIA hardware catalog
`Device CreateWithItem(DeviceItemId, DeviceItemName, DeviceName)`
- Create only the device
`Device Create(DeviceTypeId, DeviceName)`

Name	Type	Description
DeviceItemId	string	Type identifier of the device item
DeviceTypeId	string	Type identifier of the device
DeviceItemName	string	Name of the created device item
DeviceName	string	Name of the created device

See: Type identifier (Page 221)

Program code: Create device with type identifier

Modify the following code to create a device object via a type identifier:

```
DeviceComposition devices = ...;
Device device = devices.CreateWithItem("OrderNumber:6ES7 510-1DJ01-0AB0/V2.0", "PLC_1",
"NewDevice");
Device gsdDevice = devices.CreateWithItem("GSD:SIEM8139.GSD/M/4 ", "GSD Module",
"NewGsdDevice");
```

Program code: Create only the device

Modify the following code to create only the device object:

```
DeviceComposition devices = ...;
Device deviceOnly = devices.Create("System:Device.S7300", "S7300Device");
Device gsdDeviceOnly = devices.Create("GSD:SIEM8139.GSD/D", "GSD Device");
```

7.15.7 Enumerating devices

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application: Enumerating devices

The TIA Portal Openness API positions devices similar to the project navigation in TIA Portal PNV.

- Devices located as direct children of project are aggregated using the "Devices" composition of the project
- Devices located in device folders are aggregated using the "Devices" composition of the folder.

Note

Observe the AUTOHOTSPOT.

Use one of the following options to enumerate the devices of a project:

- Enumerate all devices at root level
- Enumerate all devices in groups or sub-groups
- Enumerate all devices of a project that contains no device groups
- Enumerate all devices of the ungrouped device system groups

Examples of devices that can be enumerated:

- Central station
- PB-Slave / PN-IO device
- HMI Device

Program code: Enumerating devices at root level

Modify the following program code to enumerate devices at root level:

```
private static void EnumerateDevicesInProject(Project project)
{
    DeviceComposition deviceComposition = project.Devices;
    foreach (Device device in deviceComposition)
    {
        // add code here
    }
}
```

Modify the following program code to access an individual device.

```
private static void AccessSingleDeviceByName(Project project)
{
    DeviceComposition deviceComposition = project.Devices;
    // The parameter specifies the name of the device
    Device device = deviceComposition.Find("MyDevice");
}
```

Program code: Enumerating devices in groups or sub-groups

To access devices in a group, you have to navigate to the group at first, after that to the device.

7.15 Functions on devices

Modify the following program code:

```
//Enamerate devices in groups or sub-groups
private static void EnumerateDevicesInGroups(Project project)
{
    foreach (DeviceUserGroup deviceUserGroup in project.DeviceGroups)
    {
        EnumerateDeviceUserGroup(deviceUserGroup);
    }
}
private static void EnumerateDeviceUserGroup(DeviceUserGroup deviceUserGroup)
{
    EnumerateDeviceObjects(deviceUserGroup.Devices);
    foreach (deviceUserGroup subDeviceUserGroup in deviceUserGroup.Groups)
    {
        // recursion
        EnumerateDeviceUserGroup(subDeviceUserGroup);
    }
}
private static void EnumerateDeviceObjects(DeviceComposition deviceComposition)
{
    foreach (Device device in deviceComposition)
    {
        // add code here
    }
}
```

Programm Code: Finding specific devices

Modify the following program code to find a specific device by name:

```
//Find a specific device by name
Project project = ...
Device plc1 = project.Devices.First(d => d.Name == "Mydevice");
... // Work with the device
```

Modify the following program code to find a specific device via "Find" method:

```
//Find a specific device via "Find" method
Project project = ...
Device plc1 = project.Devices.Find("MyDevice");
... // Work with the device
```

Program code: Enumerating devices of a project that contains no device groups

Modify the following program code:

```
//Enumerate all devices which are located directly under a project that contains no device
groups
Project project = ...
foreach (Device device in project.Devices)
{
    ... // Work with the devices
}
```

Program code: Enumerating all devices located in a folder

Modify the following program code:

```
//Enumerate all devices located in a folder
Project project = ...
DeviceUserGroup sortingGroup = project.DeviceGroups.Find ("Sorting");
Device plc1 = sortingGroup.Devices.First(d => d.Name == "MyStationName");
... // Work with the device
```

Program code: Enumerating devices of the ungrouped device system groups

To structure the projects, decentral devices have been put to the UngroupedDevices group. To access this group, navigate to the group at first, after that to the device.

Modify the following program code:

```
//Enumerate devices of the ungrouped device system group
Project project = ...
DeviceSystemGroup group = project.UngroupedDevicesGroup;
Device plc1 = group.Devices.First(d => d.Name == "MyStationName");
... // Work with the device
```

7.15.8 Accessing devices**Requirement**

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

Every GSD or GSDML based IO device has attributes. Some of them are used to identify the specific type of the device.

Name	Data type	Writable	Access	Description
Author	string	read/write	dynamic	
Comment	string	read/write	dynamic	
GsdName	string	read	dynamic	Name of the GSD or GSDML file.
GsdType	string	read	dynamic	Type of the hardware object. For devices the value is always "D".
GsId	string	read	dynamic	Specific identifier for the hardware object. For devices always empty.
IsGsd	bool	read		TRUE in case of a GSD device or a GSDML device
Name	string	read/write		
TypeIdentifier	string	read		

Program code: Get identification attributes

Modify the following program code to get the attributes:

```
Device device = ...;
var attributeNames = new[] {
    "GsdName", "GsdType", "GsId"
};
foreach (var attributeName in attributeNames) {
    object attributeValue = device.GetAttribute(attributeName);
}
```

Program code: Attributes

Modify the following program code to get the attributes:

```
Device device = ...;
string nameValue = device.Name;
bool isGsdValue = device.IsGsd;
```

Program code: Attributes with dynamic access

Modify the following program code to get the attributes:

```
Device device = ...;
var attributeNames = new[] {
    "GsdName", "GsdType", "GsdId"
    ;
foreach (var attributeName in attributeNames) {
    object attributeValue = device.GetAttribute(attributeName);
}
```

Specifics of GSD devices

If a device is a GSD device, it provides additional functionality. To get the GsdDevice feature, the GetService method is used.

```
GsdDevice gsdDevice = ((IEngineeringServiceProvider)deviceItem).GetService<GsdDevice>();
if (gsdDevice != null) {
    ... // work with the GSD device
};
```

Program code: Attributes of a GSD device

Modify the following program code to get the attributes:

```
Device device = ...;
GsdDevice gsdDevice = ...;
string gsdId = gsdDevice.GsdId;
string gsdName = gsdDevice.GsdName;
string gsdType = gsdDevice.GsdType;
bool isProfibus = gsdDevice.IsProfibus;
bool isProfinet = gsdDevice.IsProfinet;
```

7.15.9 Deleting a device

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Program code

Modify the following program code to delete a device:

```
Project project = ...;
Device deviceToDelete = project.UngroupedDevices.Devices.Find(".....");

// delete device
deviceToDelete.Delete();
```

7.16 Functions on device items

7.16.1 Mandatory attributes of device items

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

Every device or device item provides certain mandatory attributes which can be read and/or written. These attributes are always the same as in the TIA Portal user interface.

The following attributes are supported in TIA Portal Openness:

Attribute name	Data type	Writable	Access	Comment
Author	string	read/write	dynamic	
Classification	DeviceItemClassification	read		
Comment	string	read/write	dynamic	sometimes only read access
CommentML	MultilingualTextItem	read/write	dynamic	sometimes only read access
FirmwareVersion	string	read	dynamic	
InterfaceOperatingMode	InterfaceOperatingModes	read/write	dynamic	For device items that provides the feature NetworkInterface
InterfaceType	NetType	read/write	dynamic	For device items that provides the feature NetworkInterface
IsBuiltIn	bool	read		FALSE for objects creatable by the user
IsGsd	bool	read		TRUE, if the device description is installed via GSD/GSDML
IsPlugged	bool	read		TRUE for devices that are plugged

Attribute name	Data type	Writable	Access	Comment
Label	string	read	dynamic	For device items that provides the feature NetworkPort or NetworkInterface. If the interface or port has no label Label will be String.Empty.
LocationIdentifier	string	read/write	dynamic	
Name	string	read/write		sometimes only read access
OrderNumber	string	read/write	dynamic	sometimes only read access
PlantDesignation	string	read/write	dynamic	
PositionNumber	int	read		
TypeIdentifier	string	read		
TypeName	string	read	dynamic	The language independent type name. Optional for device items that are not manageable by the user as auto-created items or fixed sub-modules).
InstallationDate	DateTime	read/write	dynamic	
AdditionalInformation	string	read/write	dynamic	

Device item classification

Value	Description
Siemens.Engineering.HW.DeviceItemClassifications.None	No classification.
Siemens.Engineering.HW.DeviceItemClassifications.CPU	The device item is a CPU
Siemens.Engineering.HW.DeviceItemClassifications.HM	The device item is a head module.

Program code: Mandatory attributes of a device item

Modify the following program code to get the mandatory attributes of a device item:

```
DeviceItem deviceItem = ...;
string nameValue = deviceItem.Name;
string typeIdentifierValue = deviceItem.TypeIdentifier;
int positionNumberValue = deviceItem.PositionNumber;
bool isBuiltInValue = deviceItem.IsBuiltIn;
bool isPluggedValue = deviceItem.IsPlugged;
```

Program code: Mandatory attributes with dynamic access

Modify the following program code to get the attributes item with dynamic access:

```
Device device = ...;
var attributeNames = new[] {
    "TypeName", "Author", "Comment", "OrderNumber", "FirmwareVersion", "PlantDesignation",
    "LocationIdentifier"
};
foreach (var attributeName in attributeNames) {
    object attributeValue = ((IEngineeringObject)deviceItem).GetAttribute(attributeName);
}

DeviceItem deviceItem = ...;
((IEngineeringObject)deviceItem).SetAttribute("Comment", "This is a comment.");
```

7.16.2 Creating and plugging a device item

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The action `PlugNew(string typeIdentifier, string name, int positionNumber)` of `HardwareObject` is used to

- create a new device item and plug it into an existing hardware object
- create a new sub device item, e.g. a submodule, and plug it into a device item

If the action was successful it returns the created device item object, otherwise a recoverable exception will be thrown.

By using the action `CanPlugNew(string typeIdentifier, string name, int positionNumber)` you can determine if creating and plugging is possible. If executing is not possible the action returns `false`.

If the method returns `true`, the action might still fail for the following unforeseen reasons.

- a position number is already taken by another device item
- the current device item cannot be plugged at the position although it is free
- the container does not provide the position number
- the name of the device item is already taken by an existing device item in the same container

- the device item cannot be plugged into the container
- the device is online

The following table shows the needed method parameters:

Name	Type	Description
typeIdentifier	string	type identifier of the created device item
name	string	name of created device item
positionNumber	int	position number of the created device item

Program code

Modify the following program code to plug a device item into an existing hardware object:

```
HardwareObject hwObject = ...;
string typeIdentifier = ...;
string name = ...;
int positionNumber = ...;
if(hwObject.CanPlugNew(typeIdentifier, name, positionNumber))
{
    DeviceItem newPluggedDeviceItem = hwObject.PlugNew(typeIdentifier, name,
positionNumber);
}
```

Accessing module information

The TIA Portal Openness user can access information about the pluggable modules using ModuleInformationProvider object. The user can access

- the container types in which a specified module has to be plugged (ex. Device and Rack) using FindContainerTypes method.
- the available versions for a certain partly specified module using FindModuleTypes method.

Program code: Accessing ModuleInformationProvider object

```
Project project = ...;
HardwareUtilityComposition extensions = project.HwUtilities;
var result = extensions.Find("ModuleInformationProvider") as ModuleInformationProvider;
```

Program code: Accessing container types using FindContainerTypes method

FindContainerTypes method returns the container types of a given module. The module is specified using typeIdentifier parameter. The resulting list contains the TypeIdentifiers of all container types of the requested type. Usually this includes a Device and a Rack and the containers are given in their order of the hierarchy in the project, beginning with the Device.

Name of the parameter	Type	Description
typeIdentifier	string	type identifier of a device item.

NOTICE

This method works only for the modules visible in the network view.

This method works only for modules, and not for sub-modules.

```
string typeIdentifier = ...;
string[] containerTypes = moduleInformationProvider.FindContainerTypes(typeIdentifier);
```

Program code: Accessing versions using FindModuleTypes method

FindModuleTypes method returns all possible versions of a hardware object using partial type identifier of the device item. This method returns a list of strings. Each string will be the complete TypeIdentifier of a possible match for the partial TypeIdentifier.

A valid partial type identifier must contain only complete parts, where each complete part is separated by "/" in the type identifier. Wildcards or incomplete parts are not supported. Also the following constraints with respect to the minimal number of specified parts must be observed:

- OrderNumber: at least one part. Ex. OrderNumber:6ES7 317-2EK14-0AB0
- GSD: at least two parts. Ex. GSD:SI05816A.GSD/M
- System: at least one part. Ex. System:Rack.ET200SP

Name of the parameter	Type	Description
partialTypeIdentifier	string	partial type identifier of a device item

```
string partialTypeIdentifier = ...;
string[] moduleTypes = moduleInformationProvider.FindModuleTypes(partialTypeIdentifier);
```

Program code: Accessing plug locations using GetPlugLocations method

GetPlugLocations method returns the information about slots such as plug location, position number (designation of a slot), and available slots for the hardware object.

The class `PlugLocation` has the following properties.

Name of the property	Type	Description
<code>PositionNumber</code>	<code>int</code>	The position number of the free slot
<code>Label</code>	<code>string</code>	The label of the free slot

- In case no "label" exists for a certain position number, the string representation of the position number is used.
- `PlugLocation` objects are provided only for free slots.

```
HardwareObject hardwareObject = ...;
IList<PlugLocation> result = hardwareObject.GetPlugLocations();
foreach (PlugLocation item in result)
{
    Console.WriteLine("{0} - {1}", item.PositionNumber, item.Label);
}
```

7.16.3 Moving device items into another slot

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The action `PlugMove(DeviceItem deviceltem, int positionNumber)` of `HardwareObject` is used to move an existing device item and plug it to an existing hardware object. The method `PlugMove` inserts the device items where the module was unable to plug in the UI. In these cases, there `PlugMove` action completes with compile errors.

The action `CanPlugMove(DeviceItem deviceltem, int positionNumber)` is used to determine possibility of movement. If the movement is not possible, `CanPlugMove` returns false. If the method returns true, the action might still fail for the following unforeseen reasons.

- a position number is already taken by another device item
- the current device item cannot be plugged at the position although it is free
- the container does not provide the position number
- the name of the device item is already taken by an existing device item in the same container
- the device item cannot be plugged into the container
- the device item cannot be plugged by the user

7.16 Functions on device items

- the device item cannot be removed by the user
- the device is online

Program code

Modify the following program code:

```
HardwareObject hwObject = ...;
DeviceItem deviceItemToMove = ...;
int positionNumber = ...;
if(hwObject.CanPlugMove(deviceItemToMove, positionNumber)
{
    DeviceItem movedDeviceItem = hwObject.PlugMove(deviceItemToMove, positionNumber);
}
```

7.16.4 Copying a device item

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

Use the action `PlugCopy(DeviceItem deviceItem, int positionNumber)` of `HardwareObject` to copy a device within a project and to plug it into an existing hardware. In rare cases the method `PlugCopy` might work where it is not possible to plug a module in the UI. In this case compile errors will occur after the copy. When `PlugCopy` was successful it returns the copy of the device item object, otherwise a recoverable exception is thrown.

Possible reasons for a failed action:

- a position number is already used by another device item
- the current device item cannot be plugged at the position although it is free
- the container does not provide the position number
- the name of the device item is already used by an existing device item in the same container
- the device item cannot be plugged into the container
- the device item cannot be plugged in the UI
- ...

Use the action `CanPlugCopy(DeviceItem deviceItem, int positionNumber)` to determine if copying should be possible. When it is not possible to execute the copy action

CanPlugCopy returns false. However if the method returns true the action might still fail for unforeseen reasons.

Name of the parameter	Type	Description
deviceItem	DeviceItem	Device item to copy
positionNumber	int	position number for the copy of the device item

Program code

Modify the following program code:

```
HardwareObject hwObject = ...;
DeviceItem deviceItemToCopy = ...;
int positionNumber = ...;
if(hwObject.CanPlugCopy(deviceItemToCopy, positionNumber))
{
    DeviceItem copiedDeviceItem = hwObject.PlugCopy(deviceItemToCopy, positionNumber);
```

7.16.5 Deleting a device item

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Program code

Modify the following program code to delete a device item:

```
Project project = ...;
var device = project.UngroupedDevicesGroup.Devices.Find(".....");
var deviceItem = device.DeviceItems.First();

// delete device item
deviceItem.Delete();
```

7.16.6 Enumerate device items

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

To get to a device item use the `HardwareObject`. The items of a hardware objects are, what the user of the TIA Portal sees as being plugged into the hardware object:

- a rack which resides in a device
- a module which resides in a rack
- a sub module which resides in a module
- a sub module which resides in a sub module

Note

You can find more detailed information on this topic in the section AUTOHOTSPOT.

Program code: Enumerating device items of a device

Modify the following program code to enumerate device items of a hardware object:

```
private static void EnumerateDeviceItems(HardwareObject hardwareObject)
{
    foreach (DeviceItem deviceItem in hardwareObject.Items)
    {
        // add code here
    }
}
```

Program code: Enumerating with composition hierarchy

Modify the following program code if you want to enumerate the device items of a device by means of the composition hierarchy:

```
//Enumerates devices using an composition
private static void EnumerateDeviceItems(Device device)
{
    DeviceItemComposition deviceItemComposition = device.DeviceItems;
    foreach (DeviceItem deviceItem in deviceItemComposition)
    {
        // add code here
    }
}
```

Program code: Enumerating devices items using an association

Modify the following program code to enumerate the device items using an association:

```
//Enumerates devices using an association
private static void EnumerateDeviceItemsWithAssociation(Device device)
{
    DeviceItemAssociation deviceItemAssociation = device.Items;
    foreach (DeviceItem deviceItem in deviceItemAssociation)
    {
        // add code here
    }
}
```

7.16.7 Accessing device items

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application: Accessing device items

To access objects of the type "DeviceItem" use the following attributes:

- Name (string): the name of the device item
- Container (HardwareObject): the container into which the device item is plugged

7.16 Functions on device items

Name	Data type	Writeable	Access	Description
Author	string	read/write	dynamic	
Comment	string	read/write	dynamic	
FirmwareVersion	string	read	dynamic	Only for head modules
GsdName	string	read	dynamic	Name of the GSD file.
GsdType	string	read	dynamic	Type of the hardware object. For devices the value is always "D".
GsdlId	string	read	dynamic	Specific identifier for the hardware object. For devices always empty.
IsBuiltIn	bool	read		
IsGsd	bool	read		TRUE in case of a GSD device or a GSDML device
IsPlugged	bool	read		
IsProfibus	bool	read		
IsProfinet	bool	read		
Name	string	read/write		
OrderNumber	string	read	dynamic	Only for head modules
PositionNumber	bool	read		
TypeIdentifier	string	read		

Program code: Accessing a device item

Modify the following program code to access a device item:

```
public static DeviceItem AccessDeviceItemFromDevice(Device device)
{
    DeviceItem deviceItem = device.DeviceItems[0];
    return deviceItem;
}
```

Program code: Accessing a device item of a device item

Modify the following program code to access a device item of a device item:

```
public static DeviceItem AccessDeviceItemFromDeviceItem(DeviceItem deviceItem)
{
    DeviceItem subDeviceItem = deviceItem.DeviceItems[0];
    return subDeviceItem;
}
```

Program code: Navigating to the container of a device item

Modify the following program code to navigate back to the container of a device item via the "Container" attribute of DeviceItem:

```
DeviceItem deviceItem = ...;
HardwareObject container = deviceItem.Container;
```

Program code: Get identification attributes

Modify the following program code to get the attributes:

```
Device device = ...;
var attributeNames = new[] {
    "GsdName", "GsdType", "GsdId" };
foreach (var attributeName in attributeNames) {
    object attributeValue = ((IEngineeringObject)deviceItem).GetAttribute(attributeName);
}
```

Program code: Get attributes

Modify the following program code to get the attributes:

```
DeviceItem deviceItem = ...;
GsdDeviceItem gsdDeviceItem =
((IEngineeringServiceProvider)deviceItem).GetService<GsdDeviceItem>();

string gsdName = gsdDeviceItem.GsdName;
string gsdType = gsdDeviceItem.GsdType;
string gsdId = gsdDeviceItem.GsdId;
bool isProfinet = gsdDeviceItem.IsProfinet;
bool isProfibus = gsdDeviceItem.IsProfibus;;
```

Program code: Get attributes with dynamic access

Modify the following program code to get the attributes:

```
DeviceItem deviceItem = ...;
GsdDeviceItem gsdDeviceItem =
((IEngineeringServiceProvider)deviceItem).GetService<GsdDeviceItem>();

var attributeNames = new[] {
    "TypeName", "Author", "Comment", ...
};
foreach (var attributeName in attributeNames) {
    object attributeValue =
((IEngineeringObject)gsdDeviceItem).GetAttribute(attributeName);
}
```

Program code: Set attributes

Modify the following program code to set the attributes:

```
DeviceItem deviceItem = ...;
((IEngineeringObject)deviceItem).SetAttribute("Comment", "This is a comment.");
```

Program code: Get prm data of a head module

Modify the following program code to get the prm data:

```
DeviceItem deviceItem = ...;
GsdDeviceItem gsdDeviceItem =
((IEngineeringServiceProvider)deviceItem).GetService<GsdDeviceItem>();

int dsNumber = 0;           // For Profibus GSDs, dataset number zero must be used!
int byteOffset = 0;
int lengthInBytes = 5;

// read complete data set:
byte[] prmDataComplete = gsdDeviceItem.GetPrmData(dsNumber, byteOffset, lengthInBytes);

// read partial data set (only second byte):
byteOffset = 1;
lengthInBytes = 1;
byte[] prmDataPartial = gsdDeviceItem.GetPrmData(dsNumber, byteOffset, lengthInBytes);
```

Program code: Set prm data of a head module

Modify the following program code to get the prm data:

```
DeviceItem deviceItem = ...;
GsdDeviceItem gsdDeviceItem =
((IEngineeringServiceProvider)deviceItem).GetService<GsdDeviceItem>();

// The parameters byteOffset and the length of the byte array prmData define the range
// within the
// dataset which is written to.
// For Profibus GSDs, dataset number zero must be used!

// Change the highlighted bytes 2-4 from 0x0 to 0x1
// to write only the first two bytes: byte[] prmData = {0x05, 0x21};

int dsNumber = 0;
int byteOffset = 0;
byte[] prmData = {0x05, 0x21, 0x01, 0x01, 0x01};

gsdDeviceItem.SetPrmData(dsNumber, byteOffset, prmData);
```

7.16.8 Accessing device item as interface

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

If a device item is an interface, it provides additional functionalities over a simple device item. Using this interface, the user can access the nodes and operation mode of the interface. Due to this functionality, the device item can be used as IoDevice (Slave) or IoController (Master) by accessing the NetworkInterface feature (a specific Service of the device item).

The properties of the interface are accessed using enum InterfaceOperatingModes.

Value	Description
InterfaceOperatingModes.None	Default
InterfaceOperatingModes.IoDevice	Interface operation mode "IoDevice" (Slave).
InterfaceOperatingModes.IoController	Interface operation mode "IoController" (Master).
InterfaceOperatingModes.IoDevice or InterfaceOperatingModes.IoController	Interface operation mode both of the above.

Program code: Accessing the network interface feature

Modify the following program code to get the network interface feature

```
NetworkInterface itf =
((IEngineeringServiceProvider)deviceItem).GetService<NetworkInterface>();
if (itf != null)
{
    ... // work with the interface
}

//Accessing nodes and operating mode
NodeComposition nodes = itf.Nodes;
InterfaceOperationModes mode = itf.InterfaceOperatingMode;

//Accessing the type of interface
NetType itfType = itf.InterfaceType;

//Modifying the operating mode and interface type
itf.InterfaceOperatingMode = InterfaceOperatingModes.IoDevice;
itf.InterfaceType = NetType.Profibus

//Accessing the ports linked to an interface.
NetworkPortAssociation nodes = itf.Ports;
```

7.16.9 Accessing attributes of an I/O device interface

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- For writing access, the PLC is offline.

Application

You can use the TIA Portal Openness API interface to get or set attributes for IRT and isochronous mode on the I/O device interface.

Access to the interface of an I/O controller

The following attributes can be accessed to the interface of an I/O controller. The controller has to be the sync master:

Attribute name	Data type	Writeable	Access	Description
PnSendClock	Int64	r/w	Dynamic attribute	Send clock in nanoseconds

Access to the interface of an I/O system

The following attributes can be accessed to the interface of an I/O system. The Ti/To values can be used by all modules and sub modules which belong to the I/O system.

Attribute name	Data type	Writable	Access
IsochronousTiToAutoCalculation	BOOL	r/w	Dynamic attribute
IsochronousTi	DOUBLE	r/w	Dynamic attribute
IsochronousTo	DOUBLE	r/w	Dynamic attribute

Access to the interface of an I/O device

The following attributes can be accessed to the interface of an I/O device. The Ti/To values can be used by all modules and submodules which belong to the I/O system.

Attribute name	Data type	Writable	Access
IsochronousMode	BOOL	r/w	Dynamic attribute
IsochronousTiToCalculationMode	IsochronousTiToCalculationMode	r/w	Dynamic attribute
IsochronousTi	DOUBLE	r/w	Dynamic attribute
IsochronousTo	DOUBLE	r/w	Dynamic attribute

The following ENUM values are provided for the attribute IsochronousTiToCalculationMode:

Value	Description
IsochronousTiToCalculationMode.None	
IsochronousTiToCalculationMode.FromOB	Ti/To values of the OB (configured at the IoSystem) are used.
IsochronousTiToCalculationMode.FromSubnet	This value is not used by PROFINET interfaces.
IsochronousTiToCalculationMode.AutomaticMinimum	Ti/To values are calculated automatically for the IO Device.
IsochronousTiToCalculationMode.Manual	The user can enter Ti/To values for this IO Device manually.

Program code: Get or set attributes of an I/O device interface

Modify the following program code to access the send clock value:

```
DeviceItem pnInterface = ...;
// read attribute
long attributeValue = (long)pnInterface.GetAttribute("PnSendClock");
// write attribute
long sendClock = 2000000;
pnInterface.SetAttribute("PnSendClock", sendClock);
```

7.16 Functions on device items

Modify the following program code to access the Ti/To values of OB:

```
IoSystem ioSystem = ...;
bool titoAutoCalculation = (bool)ioSystem.GetAttribute("IsochronousTiToAutoCalculation");
ioSystem.SetAttribute("IsochronousTiToAutoCalculation", true);
```

Modify the following program code to access the isochronous setting of an I/O device interface:

```
DeviceItem pnInterface = ...;
bool isochronousMode = (bool)pnInterface.GetAttribute("IsochronousMode");
pnInterface.SetAttribute("IsochronousMode", true);
```

7.16.10 Accessing attributes of IoController

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- For writing access, the PLC is offline.

Application

You can use the TIA Portal Openness API interface to get or set attributes for IoController. The following attributes are only available at PROFINET IoController (located below a Profinet interface). If the user can modify an attribute in UI, then the user can set the corresponding attribute through TIA Portal Openness.

Attribute name	Data type	Type	Access	Description
SyncRole	SyncRole	Read-write	Dynamic attribute	
PnDeviceNumber	int	Read-only	Dynamic attribute	In TIA Portal UI, this property is located at the ethernet node (PROFINET section)

The Synchronization role property is available in PROFINET interface of the TIA Portal UI. The Enum SyncRole has the following values.

Enum value	Numerical value
SyncRole.NotSynchronized	0
SyncRole.SyncMaster	1
SyncRole.SyncSlave	2
SyncRole.RedundantSyncMaster	4

Program code: Setting attributes of IoController

```
IoController ioController= ...;
SyncRole syncRole = (SyncRole)((IEngineeringObject)ioController).GetAttribute("SyncRole");
((IEngineeringObject)ioController).SetAttribute("SyncRole", SyncRole.SyncMaster);
```

7.16.11 Accessing attributes of IoConnector

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- For writing access, the PLC is offline.

Application

You can use the TIA Portal Openness API interface to get or set attributes for IoConnector. The following attributes are only available at PROFINET IoController (located below a Profinet interface). If the user can modify an attribute in UI, then the user can set the corresponding attribute through TIA Portal Openness.

There are four types of attributes such as update time attributes, watchdog time attributues, synchronization attributes and , device number attributes.

Update time attributes

The update time attributes are given below.

Attribute name	Data type	Type	Access	Description
PnUpdateTimeAuto-Calculation	Boolean	Read-write	Dynamic attribute	If this attribute is true, then the update time is calculated automatically.
PnUpdateTime	Int64	Read-write	Dynamic attribute	Update time is measured in nano seconds.
PnUpdateTimeAdapt-	Boolean	Read-write	Dynamic attribute	

Watchdog time attributes

The watchdog time attributes are given below.

Attribute name	Data type	Type	Access	Description
PnWatchdogFactor	Int32	Read-write	Dynamic attribute	
PnWatchdogTime	Int64	Read-only	Dynamic attribute	Watchdog time is measured in nano seconds.

Synchronization attributes

The synchronization attributes are given below.

Attribute name	Data type	Type	Access	Description
RtClass	RtClass	Read-write	Dynamic attribute	
SyncRole	SyncRole	Read-only	Dynamic attribute	

The Enum RtClass has following values.

Enum value	Numerical value
RtClass.None	0
RtClass.RT	1
RtClass.IRT	2

The Enum SyncRole has following values.

Enum value	Numerical value
SyncRole.NotSynchronized	0
SyncRole.SyncMaster	1
SyncRole.SyncSlave	2
SyncRole.RedundantSyncMaster	4

Device number attributes

The device number attributes are given below.

Attribute name	Data type	Type	Access	Description
PnDeviceNumber	int	Read-Write	Dynamic attribute	Indicates the device number.

Program code: Getting and setting attributes of IoConnector

```
IoConnector connector = ...

var attributeNames = new[] {
    "PnUpdateTimeAutoCalculation", "PnUpdateTime", "PnUpdateTimeAdaption", "PnWatchdogFactor",
    "PnWatchdogTime", "RtClass", "SyncRole"
};

foreach (var attributeName in attributeNames)
{
    object attributeValue = ((IEngineeringObject)connector).GetAttribute(attributeName);
}

connector.SetAttribute("PnUpdateTimeAutoCalculation", true);
```

See also

[Opening a project \(Page 106\)](#)

7.16.12 Accessing address controller

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)

Application

If a device item is an address controller, it provides additional functionality. To access the registered addresses of the address controller the role AddressController is used.

Program code: Get the address controller

Modify the following program code to get the address controller role:

```
AddressController addressController =  
((IEngineeringServiceProvider)deviceItem).GetService<AddressController>();  
if (addressController != null)  
{  
    ... // work with the address controller  
}
```

Attributes of an address controller

The attributes of an address controller are:

- RegisteredAddresses

Modify the following program code to get the attributes of an address controller:

```
AddressController addressController = ...;  
foreach (Address registeredAddress in addressController.RegisteredAddresses)  
{  
    ...  
}
```

7.16.13 Accessing addresses

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

Address objects are acquired via the composition link Addresses of a device item. The attribute Addresses returns a collection of type AddressComposition which can be enumerated.

Program code: Get an address of a device item

To get the address of a device item modify the following program code:

```
AddressComposition addresses = deviceItem.Addresses;
foreach(Address address in addresses)
{
    // work with the address
}
```

Program code: Get an address of an io controller

To get the address of an io controller modify the following program code:

```
AddressComposition addresses = ioController.Addresses;
foreach(Address address in addresses)
{
    // work with the address
}
```

Attributes

An address supports the following attributes:

Attribute name	Data type	Writable	Access	Comment
AddressControllers	AddressControllerAssociation	read		
Context	enum: AddressContext	read	dynamic	only for diagnosis addresses and for special device items
IoType	enum: AddressIoType	read		
StartAdress	Int32	read/write		
Length	Int32	read		

Value	Description
AddressIoType.Diagnosis	The type of the address io is Diagnosis.
AddressIoType.Input	The type of the address io is Input.
AddressIoType.Output	The type of the address io is Output.
AddressIoType.Substitute	The type of the address io is Substitute.
AddressIoType.None	The type of the address io is not specified.

Value	Description
AddressContext.None	The address context is not applicable.
AddressContext.Device	A device address context.
AddressContext.Head	A head address context.

Program code: Read attributes

Modify the following program code to get the attributes:

```
AddressControllerAssociation addressControllers = address.AddressControllers;  
Int32 startAddress = address.StartAddress;  
AddressIoType addressType = address.IoType;  
Int32 adressLength = address.Length;
```

Program code: Write attributes

Modify the following program code to write the attributes:

```
Address addressControllers = ...;  
  
address.StartAddress = intValueStartAddress;
```

Program code: Attributes with dynamic access

Modify the following program code to get the attributes:

```
Address address= ...;  
  
object attributeValue = ((IEngineeringObject)address).GetAttribute("Context");
```

7.16.14 Accessing hardware identifiers

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

Hardware identifier objects are acquired from the following objects:

- Device
- DeviceItem
- IoSystem

The hardware identifier is represented by the class `HwIdentifier` and is accessed via the attribute `HwIdentifiers`.

Program code: Get the hardware identifier

To make HwIdentifier available modify the following program code:

```
var hwObject = ...
foreach(HwIdentifier hardwareIdentifier in hwObject.HwIdentifiers)
{
    // Work with the HwIdentifier
}
```

Attributes of a hardware identifier

```
HwIdentifierControllerAssociation controllers = hwIdentifier.HwIdentifierControllers;
Int64 Identifier = hwIdentifier.Identifier;
```

7.16.15 Accessing hardware identifier controller

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

If a device item is an hardware identifier controller, it is possible to access the registered hardware identifiers. To access these HwIdentifierController, a specific service of the device item, is used.

Program code: Get the hardware identifier controller

To get the HwIdentifierController modify the following program code:

```
HwIdentifierController hwIdentifierController =
((IEngineeringServiceProvider)deviceItem).GetService<HwIdentifierController>();
if (hwIdentifierController != null)
{
    ... // work with the hardware identifier controller
}
```

Program code: Attributes of a hardware identifier controller

The attributes of an address controller are:

- `RegisteredHwIdentifiers`: The hardware identifier controllers where the hardware identifier is registered.

Modify the following program code to get the attributes of an address controller:

```
HwIdentifierController hwIdentifierController = ...;
HwIdentifierAssociation controllers = hwIdentifierController.RegisteredHwIdentifiers;
```

7.16.16 Accessing channels of device items

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

A channel is represented by the `Channel` class. Channels are acquired from a device item via the attribute `Channels` of the `DeviceItem` class. The attribute `Channels` returns an implementation of `ChannelComposition` which can be enumerated. If the device items has no channels, the attribute `Channels` returns an empty collection.

Mandatory attributes

A channel supports the following mandatory attributes:

Attribute name	Data type	Writable	Access	Comment
<code>IoType</code>	<code>ChannelIoType</code>	read		
<code>Type</code>	<code>ChannelType</code>	read		
<code>Number</code>	<code>Int32</code>	read		
<code>ChannelAddress</code>	<code>Int32</code>	read	dynamic	Address of the channel in bits
<code>ChannelWidth</code>	<code>UInt32</code>	read	dynamic	Width of the channel in bits

Program code: Get channels of device item

Modify the following program code to get the channels of a device item:

```
ChannelComposition channels = deviceItem.Channels
foreach(Channel channel in channels)
{
    // work with the channel
}
```

Program code: Mandatory attributes of a channel

Modify the following program code to get the channels of a device item:

```
Channel channel = ...;
int channelNumber = channel.Number;
ChannelType type = channel.Type;
ChannelIoType ioType = channel.IoType;
```

Program code: Get values of attributes with dynamic access

Modify the following program code to get the values of dynamic attributes:

```
Channel channel = ...;
Int32 channelAddress = (Int32)((IEngineeringObject)channel).GetAttribute("ChannelAddress");
UInt32 channelWidth = (UInt32)((IEngineeringObject)channel).GetAttribute("ChannelWidth");
```

Program code: Set value of a dynamic attribute

Modify the following program code to set the value of a writeable dynamic attribute:

```
Channel channel = ...;
((IEngineeringObject)channel).SetAttribute("AnAttribute", 1234);
```

7.16.17 Creating and exporting psc file

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- Opening a Project
See Opening a project (Page 106)

Application

You can use the TIA Portal Openness API to create .psc file and download a device into it. In order to create a.psc file and download a device configuration into it, you can use Export method from a service CardReaderPscProvider. The service exists in namespace Siemens.Engineering.HW.Utilities.

Program code

Modify the following program code to create and export a .psc file.

```
// preconditions
Device ipcDevice = myProject.Devices.Find("IPC427D");
FileInfo exportFile = new FileInfo(@"C:\Users\Ertan\Documents\Automation\PC system
configuration74.psc");
// get the card reader provider from hardware utilities
HardwareUtilityComposition utilities = myProject.HwUtilities;
HardwareUtility utility = utilities.Find("CardReaderPscProvider");
CardReaderPscProvider crp = (CardReaderPscProvider)utility;
// do the export
crp.Export(ipcDevice, exportFile);
```

Creating and opening a .psc file with separate commands are not supported. In case you give already existing .psc file as a parameter, the file will not be overwritten and exception will be thrown. If it is not an existing file, then this .psc file will be created and finally the download will be made.

You have to make sure that the project can be compiled without problems. Otherwise, exception will be thrown.

Note

Because of safety reasons, Export operation is not supported for f-activated devices (an exception will be thrown during export) in V15.1

7.16.18 Connection handling for extension racks

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)

Application

You can use the TIA Portal Openness to fetch, add, and remove extension rack connections so that you can make use of TIA Portal Openness to implement extension rack connection support during CAx export/import.

Program code

```
ImConnection imConnection = portDeviceItem.GetService<ImConnection>();  
imConnection.Connect(partnerport);  
imConnection.Disconnect();  
imConnection.GetPartnerPort();  
var imConnectionOwner = imConnection.OwnedBy;
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.17 Functions for accessing the data of an HMI device

7.17.1 Screens

7.17.1.1 Creating user-defined screen folders

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)

Program code

Modify the following program code to create a user-defined screen folder:

```
//Creates a screen folder  
private static void CreateScreenFolder(HmiTarget hmitarget)  
{  
    ScreenUserFolder myCreatedFolder =  
    hmitarget.ScreenFolder.Folders.Create("myScreenFolder");  
}
```

7.17.1.2 Deleting a screen from a folder

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

Note

You cannot delete a permanent area. A permanent area is a system screen that is always present.

Program code

Modify the following program code to delete a screen from a specific folder:

```
public static void DeleteScreenFromFolder(HmiTarget hmiTarget)
{
    ScreenUserFolder screenUserFolder =
    hmiTarget.ScreenFolder.Folders.Find("myScreenFolder");
    ScreenComposition screens = screenUserFolder.Screens;
    Screen screen = screens.Find("myScreenName");
    if (screen != null)
    {
        screen.Delete();
    }
}
```

7.17.1.3 Deleting a screen template from a folder

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- The project contains an HMI device.

Program code

Modify the following program code to delete a screen template from a specific folder:

```
private static void DeleteScreenTemplateFromFolder(HmiTarget hmiTarget)
{
    string templateName = "MyScreenTemplate";
    ScreenTemplateUserFolder folder =
hmiTarget.ScreenTemplateFolder.Folders.Find("myScreenTemplateFolder");
    ScreenTemplateComposition templates = folder.ScreenTemplates;
    ScreenTemplate template = templates.Find(templateName);
    if (template != null)
    {
        template.Delete();
    }
}
```

7.17.1.4 Deleting all screens from a folder

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

Note

You cannot delete a permanent area. A permanent area is a system screen that is always present.

Program code

Modify the following program code to delete all screens from a specific folder:

```
private static void DeleteAllScreensFromFolder(HmiTarget hmitarget)
//Deletes all screens from a user folder or a system folder
{
    ScreenUserFolder folder = hmitarget.ScreenFolder.Folders.Find("myScreenFolder");
    //or ScreenSystemFolder folder = hmitarget.ScreenFolder;
    ScreenComposition screens = folder.Screens;
    List<Screen> list = new List<Screen>();
    foreach(Screen screen in screens)
    {
        list.Add(screen);
    }
    foreach (Screen screen in list)
    {
        screen.Delete();
    }
}
```

7.17.2 Cycles

7.17.2.1 Deleting a cycle

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- The project contains an HMI device.

Application

You cannot delete standard cycles.

You can identify whether cycles have actually been deleted based on the composition in the object model (composition count) of the respective cycle. It is no longer possible to access these cycles.

Program code

Modify the following program code to delete a cycle from an HMI device:

```
public static void DeleteCycle(HmiTarget hmiTarget)
{
    CycleComposition cycles = hmiTarget.Cycles;
    Cycle cycle = cycles.Find("myCycle");
    cycle.Delete();
}
```

7.17.3 Text lists

7.17.3.1 Deleting a text list

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- The project contains an HMI device.

Program code

Modify the following program code to delete a selected text list and all associated list entries from an HMI device:

```
public static void DeleteTextList(HmiTarget hmiTarget)
{
    TextListComposition textLists = hmiTarget.TextLists;
    TextList textList = textLists.Find("myTextList");
    textList.Delete();
}
```

7.17.4 Graphic lists

7.17.4.1 Deleting a graphic list

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- The project contains an HMI device.

Program code

Modify the following program code to delete a selected graphic list and all associated list entries from an HMI device:

```
private static void DeleteGraphicList(HmiTarget hmiTarget)
{
    GraphicListComposition graphicLists = hmiTarget.GraphicLists;
    GraphicList graphicList = graphicLists.Find("myGraphicList");
    graphicList.Delete();
}
```

7.17.5 Connections

7.17.5.1 Deleting a connection

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- The project contains an HMI device.

Program code

Modify the following program code to delete a selected communication connection from an HMI device:

```
private static void DeleteConnection(HmiTarget hmiTarget)
{
    ConnectionComposition connections = hmiTarget.Connections;
    Connection connection = connections.Find("HMI_connection_1");
    connection.Delete();
}
```

7.17.6 Tag table

7.17.6.1 Creating user-defined folders for HMI tags

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Program code

Modify the following program code to create a user-defined folder for HMI tags:

```
private static void CreateUserfolderForHMITags(HmiTarget hmitarget)
// Creates an HMI tag user folder
{
    TagSystemFolder folder = hmitarget.TagFolder;
    TagUserFolder myCreatedFolder = folder.Folders.Create("MySubFolder");
}
```

7.17.6.2 Enumerating tags of an HMI tag table

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Program code

Modify the following program code to enumerate all tags of an HMI tag table:

```
private static void EnumerateTagsInTagtable(HmiTarget hmitarget)
// //Enumerates all tags of a tag table
{
    TagTable table = hmitarget.TagFolder.TagTables.Find("MyTagtable");
    // Alternatively, you can access the default tag table:
    // TagTable defaulttable = hmitarget.TagFolder.DefaultTagTable;

    TagComposition tagComposition = table.Tags;
    foreach (Tag tag in tagComposition)
    {
        // Add your code here
    }
}
```

7.17.6.3 Deleting an individual tag from an HMI tag table

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Program code

Modify the following program code to delete a specific tag from an HMI tag table:

```
private static void DeleteATag(HmiTarget hmiTarget)
{
    string tagName = "MyTag";
    TagTable defaultTagTable = hmiTarget.TagFolder.DefaultTagTable;
    TagComposition tags = defaultTagTable.Tags;
    Tag tag = tags.Find(tagName);
    tag.Delete();
}
```

7.17.6.4 Deleting a tag table from a folder

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- The project contains an HMI device.

Application

You cannot delete the default tag table

Program code

Modify the following program code:

```
// Delete a tag table from a specific folder
private static void DeleteTagTable(HmiTarget hmiTarget)
{
    string tableName = "myTagTable";
    TagSystemFolder tagSystemFolder = hmiTarget.TagFolder;
    TagTableComposition tagTables = tagSystemFolder.TagTables;
    TagTable tagTable = tagTables.Find(tableName);
    tagTable.Delete();
}
```

7.17.7 VB scripts

7.17.7.1 Creating user-defined script folders

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Program code

Modify the following program code to create a user-defined script subfolder below a system folder or another user-defined folder:

```
private static void CreateFolderInScriptfolder(HmiTarget hmitarget)
//Creates a script user subfolderVBScriptSystemFolder
{
    VBScriptSystemFolder vbScriptFolder = hmitarget.VBScriptFolder;
    VBScriptUserFolderComposition vbScriptFolders = vbScriptFolder.Folders;
    VBScriptUserFolder vbScriptSubFolder = vbScriptFolder.Folders.Create("mySubfolder");
}
```

7.17.7.2 Deleting a VB script from a folder

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- The project contains an HMI device.

Program code

Modify the following program code to delete a VB script from a specific folder:

```
//Deletes a vbscript from a script folderVBScriptSystemFolder
private static void DeleteVBScriptFromScriptFolder(HmiTarget hmitarget)
{
    VBScriptUserFolder vbscriptfolder =
hmitarget.VBScriptFolder.Folders.Find("MyScriptFolder");
    var vbScripts = vbscriptfolder.VBScripts;
    if (null != vbScripts)
    {
        var vbScript = vbScripts.Find("MyScript");
        vbScript.Delete();
    }
}
```

7.17.8 Deleting a user-defined folder of an HMI device

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Program code

Modify the following program code to delete an user-defined folder of an HMI device:

```
HmiTarget hmiTarget = ...;
ScreenUserFolder screenUserGroup = hmiTarget.ScreenFolder.Folders.Find("MyUserFolder");
screenUserGroup.Delete();
```

7.18 Functions for accessing the data of an HMI device (Unified)

7.18.1 Object list

Introduction

The following tables show the available objects and indicate whether these objects are supported by TIA Portal Openness.

Objects

You can control the following project data for WinCC Unified device:

Object	WinCC Scada Runtime Unified
Analog Alarms	Yes
Discrete Alarms	Yes
Alarm Classes	Yes
Data Logs	Yes
Alarm Logs	Yes
Logging tags	Yes
Tag	Yes
Connections	Yes
Device runtime setting	Yes
Screens and Screen items	Yes

7.18 Functions for accessing the data of an HMI device (Unified)

Object	WinCC Scada Runtime Unified
Plant View	Yes
Plant Object	Yes
Plant View Node	Yes
Plant Object Interface	Yes
Plant Object Interface Member	Yes
Plant Object Logging Tag	Yes

7.18.2 HMI Unified Software object

Introduction

You require an HMI Software object to have access to other elements like tags, connections, alarms and screens.

Requirements

- The TIA Portal Openness application is connected to the TIA Portal
Connecting to the TIA Portal (Page 76)
- A project is open
Opening a project (Page 106)

Program code

Modify the following program code to access HMI Software object:

```
private HmiSoftware GetHmiSoftware()
{
    HmiSoftware hmiSoftware = null;
    Project project = null;
    IList<TiaPortalProcess> tiaProcessList = TiaPortal.GetProcesses();
    //If no TIA Application instance is running, then following using statement will start it or
    //if already running then will attach to it.
    TiaPortal tiaApp = tiaProcessList.Count > 0? tiaProcessList[0].Attach(): new
    TiaPortal(TiaPortalMode.WithUserInterface);
    //If there is device projects are present in given TIA instance, then take first one to work
    upon.
    if (tiaApp.Projects.Count > 0)
        project = tiaApp.Projects[0];
    else
    {
        //if there is no device project in given TIA instance, open the existing project.
        FileInfo file = new FileInfo(@"d:\Automation\Project1\Project1.apx");
        if (!file.Exists)
            throw new FileNotFoundException("Project1.apx not found");
        project = tiaApp.Projects?.Open(file);
    }
    //After getting project, get the object representing UA software HMI device.
    var devices = project.Devices;
    if (devices != null)
    {
        var device = devices[0];
        var deviceItems = device.DeviceItems;
        if (deviceItems != null)
        {
            foreach (DeviceItem deviceItem in deviceItems)
            {
                SoftwareContainer softwareContainer = deviceItem.GetService<SoftwareContainer>();
                hmiSoftware = softwareContainer?.Software as HmiSoftware;
            }
        }
    }
}
return hmiSoftware;
}
```

Note

In the above program code, the extension .apx refers to the installed version of TIA Portal.

7.18.3 Querying errors

Introduction

You can use TIA Portal Openness to get error information about properties for the following WinCC objects:

- Analog Alarm
- Discrete Alarm
- Alarm Class
- Data Log
- Alarm Log
- Tag
- Connection
- Logging Tag
- Device runtime settings
- Screens and Screen items
- Plant View
- Plant Object
- Plant View Node
- Plant Object Interface
- Plant Object Interface Member
- Plant Object Logging Tag

Note

This feature is not applicable to System Tag and Tag Table as they don't have any property which can be in error state.

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76).
- A project is open.
See Opening a project (Page 106).
- Access to HMI Unified Software object
See HMI Unified Software object (Page 272)

Program code

Modify the following program code to query errors for tag, alarms, runtime setting and screens:

7.18 Functions for accessing the data of an HMI device (Unified)

```
private void ValidateMultipleHmiObjects()
{
    HmiSoftware hmiSoftware = GetHmiSoftware();
    IList<IValidator> objectsToValidate = new List<IValidator>();
    //Put tags in validation list.
    HmiTagComposition hmiTags = hmiSoftware.HmiTags;
    foreach (HmiTag hmiTag in hmiTags)
    {
        objectsToValidate.Add(hmiTag);
    }
    //Put alarms in validation list.
    DiscreteAlarmComposition discreteAlarms = hmiSoftware.DiscreteAlarms;
    foreach (DiscreteAlarm discreteAlarm in discreteAlarms)
    {
        objectsToValidate.Add(discreteAlarm);
    }
    //Put RuntimeSettings in validation list.
    objectsToValidate.Add(hmiSoftware.RuntimeSettings);
    foreach (IValidator validator in objectsToValidate)
    {
        IList<HmiValidationResult> validationResult = validator.Validate();
        if (validationResult != null && validationResult.Count > 0)
        {
            foreach (HmiValidationResult propertyErrors in validationResult)
            {
                //browse error for different property
                string propertyName = propertyErrors.PropertyName;
                foreach (string propertyError in propertyErrors.Errors)
                {
                    //work with property errors
                }
            }
        }
    }
    //Put screens in validation list.
    foreach (var screen in hmiSoftware.Screens)
    {
        objectsToValidate.Add(screen);
    }
    foreach (IValidator validator in objectsToValidate)
    {
        IList<HmiValidationResult> errors = validator.Validate();
        if (errors != null && errors.Count > 0)
        {
            foreach (var errornotification in errors)
            {
                //browse error for different property
                var propName = errornotification.PropertyName;
                foreach (var errormessage in errornotification.Errors)
                {
                    Console.WriteLine(errormessage);
                }
            }
        }
    }
}
```

```
//Put screenitems in validation list.  
foreach (var screenitem in ScreenObject.ScreenItems)  
{  
    objectsToValidate.Add(screenitem);  
}  
foreach (IValidator validator in objectsToValidate)  
{  
    IList<HmiValidationResult> errors = validator.Validate();  
    if (errors != null && errors.Count > 0)  
    {  
        foreach (var errornotification in errors)  
        {  
            //browse error for different property  
            var propName = errornotification.PropertyName;  
            foreach (var errormessage in errornotification.Errors)  
            {  
                Console.WriteLine(errormessage);  
            }  
        }  
    }  
}
```

See also

[HMI Unified Software object \(Page 272\)](#)

7.18.4 Alarms

7.18.4.1 Working with analog alarms

Introduction

You can perform the following tasks with analog alarms while using TIA Portal Openness:

- Creating analog alarms
- Enumerating analog alarms
- Deleting analog alarms
- Accessing analog alarm properties

Properties

The following properties are supported in analog alarm:

Property Name	Data Type	Description	Access
EventText	MultilingualText	Specifies event/alarm text of alarm	R/W
Id	uint32	Specify alarm number to identify alarm. This is unique value for each analog alarm	R/W
AlarmClass	String	Specifies alarm class	R/W
Name	String	Specifies alarm name	R/W
Priority	byte	Specifies alarm Priority	R/W
Origin	String	Specifies origin of alarm	R/W
Area	String	Specifies area of alarm	R/W
RaisedStateTag	String	Specifies tag that trigger/raise the alarm	R/W
ConditionValue	object Supported type: Double, int16, , uint16, int32, uint32, int64, uint64, byte	Condition Value can be specified as constant value	R/W
Condition	HmiAlarmCondition	Specifies Limit Mode	R/W
InfoText	MultilingualText	Specifies Info Text	R/W
AlarmParameterTags	Object SupportedType: IList<string>	If array of 10 parameter tags which specifies alarm parameter. Each parameter take tag name as string. These parameters can be used in event text.	R/W
EventText1	MultilingualText	Event Text of HmiAnalogAlarm	R/W
EventText2	MultilingualText	Event Text of HmiAnalogAlarm	R/W
EventText3	MultilingualText	Event Text of HmiAnalogAlarm	R/W
EventText4	MultilingualText	Event Text of HmiAnalogAlarm	R/W
EventText5	MultilingualText	Event Text of HmiAnalogAlarm	R/W
EventText6	MultilingualText	Event Text of HmiAnalogAlarm	R/W
EventText7	MultilingualText	Event Text of HmiAnalogAlarm	R/W
EventText8	MultilingualText	Event Text of HmiAnalogAlarm	R/W
EventText9	MultilingualText	Event Text of HmiAnalogAlarm	R/W

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76).
- A project is open.
See Opening a project (Page 106).
- Access to the HMI Software object.
See HMI Unified Software object (Page 272)

Program code

You can modify and use the following program code examples while working with alarms.

Creating analog alarms

Modify the following program code to create analog alarms:

```
private void AnalogAlarmCreate()
{
//Create Analog Alarm
HmiSoftware hmiSoftware = GetHmiSoftware();
AnalogAlarmComposition analogAlarms = hmiSoftware.AnalogAlarms;
AnalogAlarm analogAlarm = analogAlarms.Create("Alarm_1");
}
```

Deleting analog alarms

Modify the following program code to delete analog alarm:

```
private void AnalogAlarmDelete()
{
//Delete Analog Alarm
hmiSoftware = GetHmiSoftware();
AnalogAlarmComposition analogAlarms = hmiSoftware.AnalogAlarms;
AnalogAlarm analogAlarm = analogAlarms.Find("Alarm_1");
if (analogAlarm != null)
{
analogAlarm.Delete();
}
}
```

Enumerating analog alarms

7.18 Functions for accessing the data of an HMI device (Unified)

Modify the following program code to enumerate an analog alarm:

```
private void AnalogAlarmBrowse()
{
//Browse Analog Alarms
hmiSoftware = GetHmiSoftware();
AnalogAlarmComposition analogAlarms = hmiSoftware.AnalogAlarms;
foreach (AnalogAlarm analogAlarm in analogAlarms)
{
//work with analog alarms.
}
//Other way to get alarm by using Find function on Alarm Composition. AnalogAlarm
analogAlarmObj = analogAlarms.Find("Alarm_1");
}
```

Accessing analog alarm properties

Modify the following program code to set and get analog alarm properties:

```
private void AnalogAlarmPropertiesAcess()
{
hmiSoftware = GetHmiSoftware();
AnalogAlarmComposition analogAlarms = hmiSoftware.AnalogAlarms;
AnalogAlarm analogAlarm = analogAlarms.Find("Alarm_1");
int id = analogAlarm.Id
analogAlarm.Id = 10;
int priority = analogAlarm.Priority; analogAlarm.Priority = 7;
analogAlarm.Area = "Area_1";
string area = analogAlarm.Area;
analogAlarm.Name = "NewAlarm";
string name = analogAlarm.Name;
}
```

7.18.4.2 Working with discrete alarms

Introduction

You can perform the following tasks with discrete alarms while using TIA Portal Openness:

- Creating discrete alarms
- Deleting discrete alarms
- Enumerating discrete alarms
- Accessing discrete alarm properties

Properties

The following properties are supported in discrete alarms:

Property Name	Data Type	Description	Access
EventText	MultilingualText	Specifies event/alarm text of alarm	R/W
Id	uint32	Specify alarm number to identify alarm. This is unique value for each Discrete alarm	R/W
AlarmClass	String	Specifies alarm class	R/W
Name	String	Specifies alarm name	R/W
Priority	byte	Specifies alarm Priority.	R/W
Origin	String	Specifies origin of alarm	R/W
Area	String	Specifies area of alarm	R/W
RaisedStateTag	String	Specifies tag that raise/trigger the alarm	R/W
RaisedStateTagBitNumber	uint32	Trigger bit on trigger tag	R/W
TriggerMode	HmiDiscreteAlarmTrigger-Mode	Specifies Trigger mode for discrete alarm	R/W
InfoText	MultilingualText	Specifies Info Text	R/W
AlarmParameterTags	Object	If array of 10 parameter tags which specifies alarm parameter. Each parameter take tag name as string. These parameters can be used in alarm text	R/W
EventText1	MultilingualText	Event Text of HmiDiscreteAlarm	R/W
EventText2	MultilingualText	Event Text of HmiDiscreteAlarm	R/W
EventText3	MultilingualText	Event Text of HmiDiscreteAlarm	R/W
EventText4	MultilingualText	Event Text of HmiDiscreteAlarm	R/W
EventText5	MultilingualText	Event Text of HmiDiscreteAlarm	R/W
EventText6	MultilingualText	Event Text of HmiDiscreteAlarm	R/W
EventText7	MultilingualText	Event Text of HmiDiscreteAlarm	R/W
EventText8	MultilingualText	Event Text of HmiDiscreteAlarm	R/W
EventText9	MultilingualText	Event Text of HmiDiscreteAlarm	R/W

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76).
- A project is open.
See Opening a project (Page 106).
- Access to HMI Software object.
See HMI Unified Software object (Page 272).

Program code

You can modify and use the following program code examples while working with alarms.

Creating discrete alarms

Modify the following program code to create a discrete alarm:

```
HmiSoftware hmiSoftware = ...;
DiscreteAlarmComposition discreteAlarms = hmiSoftware.DiscreteAlarms;
DiscreteAlarm discreteAlarm = discreteAlarms.Create("Alarm_1");
```

Deleting discrete alarms

Modify the following program code to delete a discrete alarm:

```
public void Delete()
//User wants to delete a discrete alarm with the alarm name Alarm_1
{
HmiSoftware hmiSoftware = ...;
DiscreteAlarmComposition discreteAlarms = hmiSoftware.DiscreteAlarms;
DiscreteAlarm discreteAlarm = discreteAlarms.Find("Alarm_1");
discreteAlarm.Delete();
}
```

Enumerating discrete alarms

Modify the following program code to enumerate a discrete alarm:

```
private void DiscreteAlarmBrowse()
{
//Browse Discrete Alarms
HmiSoftware hmiSoftware = GetHmiSoftware();
DiscreteAlarmComposition discreteAlarms = hmiSoftware.DiscreteAlarms;
foreach (DiscreteAlarm discreteAlarm in discreteAlarms)
{
//work with discrete alarms.
}
//Other way to get alarm by using Find function on Alarm Composition.
DiscreteAlarm discreteAlarmObj = discreteAlarms.Find("Alarm_1");
}
```

Accessing discrete alarm properties

Modify the following program code to access discrete alarm properties:

```
HmiSoftware hmiSoftware = ....
DiscreteAlarm discreteAlarm = hmiSoftware.DiscreteAlarms[0];

int id = discreteAlarm.Id;
discreteAlarm.Id = 10;

int priority = discreteAlarm.Priority;
discreteAlarm.Priority = 7;

discreteAlarm.area = "Aread1";
string area = discreteAlarm.Area;

discreteAlarm.Name = "NewAlarm";
string name = discreteAlarm.Name;
```

7.18.4.3 Working with alarm classes

Introduction

You can perform the following tasks with alarm classes while using TIA Portal Openness:

- Creating alarm class
- Deleting alarm classes
- Enumerating alarm classes
- Accessing properties of alarm class

Properties

The following properties are supported in alarm class:

Property Name	Data Type	Description	Access
Name	String	Name of alarm class	Depend on type of alarm class (system / user)
Priority	byte	Priority of alarm class	R/W
IsSystem	Bool	Specifies if alarm class is system provided	R
StateMachine	HmiAlarmStateMachine	Specifies state machine of alarm class	Depend on type of alarm class (system / user)
RaisedState	HmiRaisedState	Alarm Incoming or raised Status	R
RaisedState.BackColor	Color	Specifies Back color	R/W
RaisedState.TextColor	Color	Specifies Text Color	R/W
RaisedState.Floating	Bool	Specifies floating	R/W
ClearedState	HmiClearedState	Specifies Alarm coming going or cleared status	R
ClearedState.BackColor	Color	Specifies Backcolor	R/W
ClearedState.TextColor	Color	Specifies TextColor	R/W
ClearedState.Floating	Bool	Specifies floating	R/W
AcknowledgedState	HmiAcknowledgedState	Specifies Alarm incoming acknowledge or acknowledged status	R
AcknowledgedState.BackColor	Color	Specifies Backcolor	R/W
AcknowledgedState.TextColor	Color	Specifies TextColor	R/W
AcknowledgedState.Floating	Bool	Specifies floating	R/W
AcknowledgedClearedState	HmisAcknowledgedClearedState	Specifies Alarm incoming outgoing acknowledge or AcknowledgedCleared status	
AcknowledgedClearedState.BackColor	Color	Specifies Backcolor	R/W
AcknowledgedClearedState.TextColor	Color	Specifies TextColor	R/W
AcknowledgedClearedState.Floating	Bool	Specifies floating	R/W
Id	uint32	Specify id to identify alarm class. This is unique value for each Alarm class.	R
Log	System.String	Log of Alarm class	R/W
CommonAlarmClass	System.String	Common Alarm Class	R

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76).
- A project is open.
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)

Program code

You can modify and use the following program code examples while working with alarm classes.

Creating alarm classes

Modify the following program code to create alarm class:

```
private void AlarmClassCreate()
{
//Create Alarm Class
HmiSoftware hmiSoftware = GetHmiSoftware();
AlarmClassComposition alarmClasses = hmiSoftware.AlarmClasses;
AlarmClass alarmClass = alarmClasses.Create("AlarmClass_1");
}
```

Deleting alarm classes

Modify the following program code to delete alarm class:

```
public void Delete()
//User wants to delete an alarm class with the alarm name AlarmClass_1
{
HmiSoftware hmiSoftware = ...:
AlarmClassComposition alarmClasses = hmiSoftware.AlarmClasses;
AlarmClass alarmClass = alarmClasses.Find("AlarmClass_1");
alarmClass.Delete();
}
```

Enumerating alarm classes

7.18 Functions for accessing the data of an HMI device (Unified)

Modify the following program code to enumerate alarm class:

```
//User wants to enumerate all discrete alarms of device with following code
{
HmiSoftware hmiSoftware = ....;
AlarmClassComposition alarmClasses = hmiSoftware.AlarmClasses;
foreach (AlarmClass alarmClass in alarmClasses)
}
{
...
}
```

Accessing alarm class properties

Modify the following program code to access alarm class properties:

```
HmiSoftware hmiSoftware = ....;
AlarmClassComposition alarmClasses = hmiSoftware.AlarmClasses;
AlarmClass alarmClass = hmiSoftware.AlarmClasses.Find("MostCritical");

int priority = alarmClass.Priority;
alarmClass.Priority = 10;

alarmClass.Acknowledgement = StateMachine.AlarmWithDualModeAcknowledgement;
alarmClass.IncomingOutgoingAcknowledgeStatus.BackColor = Color.Red;
alarmClass.IncomingOutgoingAcknowledgeStatus.TextColor = Color.Black;
alarmClass.IncomingOutgoingAcknowledgeStatus.Flashing = true;
```

7.18.5 Logs

7.18.5.1 Working with data logs

Introduction

You can perform the following tasks with data logs while using TIA Portal Openness:

- Create data log
- Delete data log
- Enumerate data log
- Access properties of data log

Properties

The following properties are supported in data log:

Property Name	Data Type	Description	Access
Name	String	Specifies name of data log	R/W
Settings	LogSettings	Complex property. Having properties (StorageFolder, LogMaxSize and LogTimePeriod) as members which specifies setting related to log.	R
	StorageFolder	Path for storage	R/W
	LogMaxSize	Defines maximum size of log on storage Medium in units of megabytes	R/W
	LogTimePeriod	Defines maximum time period covered by log	R/W
	StorageDevice	Defines which storage device	R/W
Segment	LogSegment	Complex property. Having properties (SegmentMaxSize, SegmentStartTime and SegmentTimePeriod) as members which specifies setting related to Segment	
	SegmentMaxSize	Define maximum size of segment of log in units of megabytes	R/W
	SegmentStartTime	Define exact point in time when segment shall be started.	R/W
	SegmentTimePeriod	Maximum time period covered by segment of log.	R/W
Backup	LogBackup	Complex property. Having properties (PrimaryPath, and BackupMode) as members which specifies setting related to log backup.	R
	BackupMode	Defines back up mode	R/W
	PrimaryPath	Path for back up	R/W

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76).
- A project is open.
See Opening a project (Page 106).
- Access to HMI Software object
See HMI Unified Software object (Page 272)

Program code

You can modify and use the following program code examples while working with data logs.

Creating data log

Modify the following program code to create a data log:

```
private void DatalogCreate()
{
//Create Datalog
HmiSoftware hmiSoftware = GetHmiSoftware();
DataLogComposition dataLogs = hmiSoftware.DataLogs;
DataLog dataLog = dataLogs.Create("Datalog_1");
}
```

Deleting data log

Modify the following program code to delete a data log:

```
private void DatalogDelete()
{
//user wants to delete datalog with name "Datalog_1"
HmiSoftware hmiSoftware = GetHmiSoftware();
DataLogComposition dataLogs = hmiSoftware.DataLogs;
DataLog dataLog = dataLogs.Find("Datalog_1");
dataLog.Delete();
}
```

Enumerating data log

Modify the following program code to enumerate data log:

```
private void DatalogBrowse()
{
//Browse Datalog.
HmiSoftware hmiSoftware = GetHmiSoftware();
DataLogComposition dataLogs = hmiSoftware.DataLogs;
foreach (DataLog dataLog in dataLogs)
{
//work with data log.
}
//Other way to get data log by using Find function on datalog Composition.
DataLog dataLogObj = dataLogs.Find("Datalog_1");
}
```

Accessing properties of data log

You can modify and use the following program code to accessing data log properties:

```
private void DatalogProperties()
{
    //Set/Get Data Log properties.
    HmiSoftware hmiSoftware = GetHmiSoftware();
    DataLogComposition dataLogs = hmiSoftware.DataLogs;
    DataLog dataLog = dataLogs.Find("Datalog_1");
    string name = dataLog.Name;
    dataLog.Settings.StorageFolder = @"C:\Logs\Log";
    dataLog.Settings.LogMaxSize = Int32.MaxValue;
    dataLog.Settings.LogMaxSize = 2000;
    LogDuration duration = dataLog.Settings.LogTimePeriod;
    double durationInDouble = duration.GetDoubleLogDuration();
    string durationInString = duration.GetStringLogDuration();
    duration.Days = 7;
    duration.Hours = 23;
    duration.Minutes = 50;
    duration.Seconds = 0;
    duration.Ticks = 1;
    //Method to set log duration.
    duration.SetLogDuration(10, 12, 4, 5, 0);
    dataLog.Backup.BackupMode = BackupMode.NoBackup;
    dataLog.Backup.BackupMode = BackupMode.PrimaryPath;
    dataLog.Backup.PrimaryPath = @"C:\Logs\Backup";
    dataLog.Segment.SegmentMaxSize = 500;
    dataLog.Segment.SegmentStartTime = DateTime.Now;
    dataLog.Segment.SegmentStartTime = DateTime.Now.Date;
}
```

7.18.5.2 Working with alarm logs

Introduction

You can perform the following tasks with alarm logs while using TIA Portal Openness:

- Create alarm log
- Delete alarm log
- Enumerate alarm log
- Access properties of alarm log

Properties

The following properties are supported in alarm log:

Property Name		Data Type	Description	Access
Name		String	Specifies name of alarm log	R/W
Settings		LogSettings	Complex property. Having properties (StorageFolder, LogMaxSize and LogTimePeriod) as members which specifies setting related to log.	R
	StorageFolder	String	Path for storage	R/W
	LogMaxSize	Unsigned int	Defines maximum size of log on storage Medium in units of megabytes	R/W
	LogTimePeriod	LogDuration	Defines maximum time period covered by log	R/W
Segment		LogSegment	Complex property. Having properties (SegmentMaxSize, StartTime and SegmentTimePeriod) as members which specifies setting related to Segment	
	SegmentMaxSize	Unsigned int	Define maximum size of segment of log in units of megabytes	R/W
	SegmentStartTime	DateTime	Define exact point in time when segment shall be started.	R/W
	SegmentTimePeriod	SegmentDuration	Maximum time period covered by segment of log.	R/W
Backup		LogBackup	Complex property. Having properties (PrimaryPath, and BackupMode) as members which specifies setting related to log backup.	R
	BackupMode	HmiBackupMode	Defines back up mode	R/W
	PrimaryPath	String	Path for back up	R/W

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76).
- A project is open.
See Opening a project (Page 106).
- Access to HMI Software object
See HMI Unified Software object (Page 272)

Program code

You can modify and use the following program code examples while working with alarm logs.

Creating alarm logs

Modify the following program code to create an alarm log:

```
private void AlarmLogCreate()
{
//Create Alarmlog
HmiSoftware hmiSoftware = GetHmiSoftware();
AlarmLogComposition alarmLogs = hmiSoftware.AlarmLogs;
AlarmLog alarmLog = alarmLogs.Create("Alarmlog_1");
}
```

Deleting alarm logs

Modify the following program code to delete an alarm log:

```
private void AlarmLogDelete()
{
//User wants to delete alarm log with name "Alarmlog_1"
HmiSoftware hmiSoftware = GetHmiSoftware();
AlarmLogComposition alarmLogs = hmiSoftware.AlarmLogs;
AlarmLog alarmLog = alarmLogs.Find("Alarmlog_1");
alarmLog.Delete();
}
```

Enumerating alarm logs

Modify the following program code to enumerate an alarm log:

```
private void AlarmLogBrowse()
{
//Browse alarm log.
HmiSoftware hmiSoftware = GetHmiSoftware();
AlarmLogComposition alarmLogs = hmiSoftware.AlarmLogs;
foreach (AlarmLog alarmLog in alarmLogs)
{
//work with alarm log.
}
//Other way to get alarmlog by using Find function on alarmlog Composition.
AlarmLog alarmLogObj = alarmLogs.Find("Alarmlog_1");
}
```

Accessing properties of alarm log

You can modify and use the following program code to accessing alarm log properties:

```
private void AlarmLogProperties()
{
//Set/Get alarm log properties.
HmiSoftware hmiSoftware = GetHmiSoftware();
AlarmLogComposition alarmLogs = hmiSoftware.AlarmLogs;
AlarmLog alarmLog = alarmLogs.Find("Alarmlog_1");
string name = alarmLog.Name;
alarmLog.Settings.StorageFolder = @"C:\Logs\Log";
alarmLog.Settings.LogMaxSize = Int32.MaxValue;
alarmLog.Settings.LogMaxSize = 2000;
LogDuration duration = alarmLog.Settings.LogTimePeriod;
double durationInDouble = duration.GetDoubleLogDuration();
string durationInString = duration.GetStringLogDuration();
duration.Days = 7;
duration.Hours = 23;
duration.Minutes = 50;
duration.Seconds = 0;
duration.Ticks = 1;
//Log duration can be set by using function also.
duration.SetLogDuration(10, 12, 4, 5, 0);
alarmLog.Backup.BackupMode = BackupMode.NoBackup;
alarmLog.Backup.BackupMode = BackupMode.PrimaryPath;
alarmLog.Backup.PrimaryPath = @"C:\Logs\Backup";
alarmLog.Segment.SegmentMaxSize = 500;
alarmLog.Segment.SegmentStartTime = DateTime.Now;
alarmLog.Segment.SegmentStartTime = DateTime.Now.Date;
}
```

See also

[HMI Unified Software object \(Page 272\)](#)

7.18.5.3 Working with logging tags

Introduction

You can perform the following tasks with logging tags while using TIA Portal Openness:

- Create logging tag
- Delete logging tag
- Enumerate logging tag
- Access properties of logging tags

Properties

The following properties are supported in logging tag:

Property Name	Data Type	Description	Access
AggregationDelay	System.TimeSpan	Aggregation delay value	R/W
AggregationMode	HmiAggregationMode	Aggregation Mode value	R/W
Cycle	System.String	Logging Cycle of logging tag	R/W
CycleFactor	System.UInt32	Logging Cycle Factor vlaue	R/W
DataLog	System.String	Data log of logging tag	R/W
HighLimit	System.Object	High limit of logging tag	R/W
LimitScope	HmiLimitScope	Limit scope of logging tag	R/W
LoggingMode	HmiLoggingMode	Logging mode of logging tag	R/W
LowLimit	System. Object	Low limit of logging tag	R/W
Name	System.string	Name of logging tag	R/W
SmoothingDeltaValue	System.Double	Delta of logging tag	R/W
SmoothingMaxTime	System. TimeSpan	Maximum time of logging tag	R/W
SmoothingMinTime	System.TimeSpan	Minimum time of logging tag	R/W
SmoothingMode	HmiSmoothingMode	Smoothing mode logging tag	R/W
Source	System.string	Source Logging Tag	R/W
TriggerMode	HmiTriggerMode	TriggerMode of Logging tag	R/W
TriggerTag	System.String	TriggerTag Value	R/W
TriggerTagBitNumber	System.UInt32	TriggerTagBitNumber value	R/W

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76).
- A project is open.
See Opening a project (Page 106).
- Access to HMI Software object
See HMI Unified Software object (Page 272)

Program code

You can modify and use the following program code examples while working with logging tags.

Creating logging tags

You can modify and use the following program code to create a logging tag:

```
private void LoggingTagCreate()
{
    //Create Logging Tag for given tag.
    HmiSoftware hmiSoftware = GetHmiSoftware();
    HmiTagComposition hmiTags = hmiSoftware.HmiTags;
    //Tag_1 exists in TIA project.
    HmiTag hmiTag = hmiTags.Find("Tag_1");
    LoggingTagComposition loggingTags = hmiTag.LoggingTags;
    LoggingTag loggingTag = loggingTags.Create("LoggingTag_1");
}
```

Deleting logging tags

You can modify the following program code to delete a logging tag:

```
private void LoggingTagDelete()
{
    //user wants to delete Logging tag with name "LoggingTag_1" for tag "Tag_1";
    HmiSoftware hmiSoftware = GetHmiSoftware();
    HmiTagComposition hmiTags = hmiSoftware.HmiTags;
    //Tag_1 exists in TIA project.
    HmiTag hmiTag = hmiTags.Find("Tag_1");
    LoggingTagComposition loggingTags = hmiTag.LoggingTags;
    //LoggingTag_1 exists in TIA project.
    LoggingTag loggingTag = loggingTags.Find("LoggingTag_1");
    loggingTag.Delete();
}
```

Enumerating Logging tags

Modify the following program code to enumerate logging tag:

```
private void LoggingTagBrowse()
{
    HmiSoftware hmiSoftware = GetHmiSoftware();
    HmiTagComposition hmiTags = hmiSoftware.HmiTags;
    //Tag_1 exists in TIA project.
    HmiTag hmiTag = hmiTags.Find("Tag_1");
    LoggingTagComposition loggingTags = hmiTag.LoggingTags;
    foreach (LoggingTag loggingTag in loggingTags)
    {
        //...
    }
    //LoggingTag_1 exists in TIA project.
    LoggingTag loggingTagObj = loggingTags.Find("LoggingTag_1");
}
```

Accessing properties of logging tags

You can modify and use the following program code to accessing logging tag properties:

```
private void LoggingTagProperties()
{
    //Set/Get LoggingTag properties.
    HmiSoftware hmiSoftware = GetHmiSoftware();
    HmiTagComposition hmiTags = hmiSoftware.HmiTags;
    //Tag_1 exists in TIA project.
    HmiTag hmiTag = hmiTags.Find("Tag_1");
    LoggingTagComposition loggingTags = hmiTag.LoggingTags;
    //LoggingTag_1 exists in TIA project.
    LoggingTag loggingTag = loggingTags.Find("LoggingTag_1");

    string name = loggingTag.Name;

    SmoothingMode smoothingMode = loggingTag.SmoothingMode;
    loggingTag.SmoothingMode = SmoothingMode.SwingingDoor;

    LimitScope limitScope = loggingTag.LimitScope;
    loggingTag.LimitScope = LimitScope.WithinLimits;

    string lowLimit = loggingTag.LowLimit;
    loggingTag.LowLimit = "-32768";

    string highLimit = loggingTag.HighLimit;
    loggingTag.HighLimit = "-3";

    TimeSpan smotthingMinTime = loggingTag.SmoothingMinTime;
    TimeSpan tsMin = TimeSpan.Parse("500");
    loggingTag.SmoothingMinTime = tsMin;

    TimeSpan smotthingMaxTime = loggingTag.SmoothingMaxTime;
    TimeSpan tsMax = TimeSpan.Parse("1000");
    loggingTag.SmoothingMaxTime = tsMax;

    double smoothingDeltaValue = loggingTag.SmoothingDeltaValue;
    loggingTag.SmoothingDeltaValue = 5;

    string dataLog = loggingTag.DataLog;
    loggingTag.DataLog = "Datalog_1";
}
```

See also

[HMI Unified Software object \(Page 272\)](#)

7.18.6 Tags and tag tables

7.18.6.1 Working with tag tables

Introduction

You can perform the following tasks with tag tables while using TIA Portal Openness:

- Create tag tables
- Delete tag tables
- Enumerate tag tables
- Access tag table properties

Properties

The following properties are supported in tag table:

Property Name	Data Type	Description	Access
Name	String	Specifies alarm name	R/W
Tags	HmiTagComposition	List of tags	R

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76).
- A project is open.
See Opening a project (Page 106).

Program code

You can modify and use the following program code examples while working with tag tables.

Creating tag tables

Modify the following program code to create a tag table:

```
private void TagTableCreate()
{
//Create Tag Table
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiTagTableComposition tagTables = hmiSoftware.HmiTagTables;
HmiTagTable hmiTagTable = tagTables.Create("TagTable_1");
}
```

Deleting tag tables

Modify the following program code to delete a tag table:

```
private void TagTableDelete()
{
//User wants to delete tag table with name "TagTable_1"
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiTagTableComposition tagTables = hmiSoftware.HmiTagTables;
HmiTagTable tagTable = tagTables.Find("TagTable_1");
tagTable.Delete();
}
```

Enumerating tag tables

Modify the following program code to enumerate tag table:

```
private void TagTableBrowse()
{
//Browse Tag Tables
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiTagTableComposition tagTables = hmiSoftware.HmiTagTables;
foreach (HmiTagTable tagTable in tagTables)
{
//Work with tag table
}
//Other way to get tag table by using Find function on tag table Composition.
HmiTagTable tagTableObj = tagTables.Find("TagTable_1");
}
```

Accessing tag table properties

Modify the following program code to accessing properties of a tag table:

```
private void TagTableProperties()
{
//Set/Get Tag Table properties.
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiTagTableComposition tagTables = hmiSoftware.HmiTagTables;
HmiTagTable tagTable = tagTables.Find("TagTable_1");
tagTable.Name = "NewTagTable";
string name = tagTable.Name;
}
```

7.18.6.2 Working with HMI tags

Introduction

You can perform the following tasks with HMI tags while using TIA Portal Openness:

- Create HMI tags
- Delete HMI tags

7.18 Functions for accessing the data of an HMI device (Unified)

- Enumerate HMI tags
- Access a HMI tag
- Access HMI tag properties
- Usage of UDT datatype for tags
- Properties of member tags of UDT

Properties

The following properties are supported in HMI tag:

Property Name	Data Type	Description	Access
AccessMode	HmiAccessMode	The Hmi Tag Access Mode	R/W
AcquisitionCycle	String	The Acquisition cycle attribute	R/W
AcquisitionMode	HmiTagAcquisitionMode	Hmi Tag AcquisitionMode	R/W
Comment	MultilingualText	Get/set comment of tag	R/W
Connection	String	The Hmi Connection	R/W
DataType	String	DataType of the Tag	R/W
MaxLength	UInt32	The hmi tag DataTypeLength	R/W
DisplayName	MultilingualText	Get/set display name of tag	R/W
Address	String	The Hmi Tag Address Attribute	R/W
InitialMinValue	LowerRange	Lower limit	R
Name	String	Name of Hmi Tag	R/W
Persistent	Boolean	The Persistence attribute	R/W
PlcName	String	The Plc Name attribute	R
PlcTag	String	The Plc Tag attribute	R/W
InitialValue	Object	The Start value attribute	R/W
SubstituteValue	HmiSubstituteValue	The SubstituteValue	R
UpdateId	UInt32	The Update Id Attribute	R/W
InitialMaxValue	UpperRange	Upper limit	R
TagName	String	Tag Table Name to which tag belongs	R
HmiDataType	String	HmiDataType of the Tag	
LinearScaling	Boolean	LinearScaling	R/W
HmiStartValue	Object		R/W
HmiEndValue	Object		R/W
PlcStartTime	Object		R/W
PlcEndTime	Object		R/W

The following properties are available in LowerRange/UpperRange:

Property Name	Data Type	Description	Access
Value	Object	Value of upper/lower	R/W
ValueType	HmiLimitValueType	Get and set Value type	R/W

The following properties are available in SubstituteValue:

Property Name	Data Type	Description	Access
Value	Object	Get and set Value type	R/W
SubstituteValueUsage	Hmi SubstituteValueUsage	Describe when to use substitute value	R/W

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76).
- A project is open.
See Opening a project (Page 106).
- Access to the HMI Software object
See HMI Unified Software object (Page 272)

Program code

You can modify and use the following program code examples while working with HMI tag.

Creating HMI tags

Modify the following program code to create a HMI tag:

```
private void CreateTag()
{
    HmiSoftware hmiSoftware = GetHmiSoftware();

    //Four ways to create tag.
    //1. Create tag by accessing HmiTags property at HmiSoftware level, by using Create method
    //with two parameters.
    HmiTagComposition hmiTags1 = hmiSoftware.HmiTags;
    //TagTable1 exists in TIA project.
    HmiTag hmiTag1 = hmiTags1.Create("Tag1", "TagTable1");

    //2. Create tag by accessing HmiTags property at HmiSoftware level, by using Create method
    //with single parameter.
    HmiTag hmiTag2 = hmiTags1.Create("Tag2");
    //Tags will be created in default tag table.

    //3. Creation of tag by accessing HmiTags property at Tag Table, by using Create method with
    //two parameters.
    //Tag creation will fail and it will result in recoverable exception,
    HmiTagComposition hmiTags2 = hmiSoftware.HmiTagTables.Find("Default tag table").HmiTags;
    HmiTag hmiTag3 = hmiTags2.Create("Tag_3", "TableTableName");

    //4. Create tag by accessing HmiTags property at Tag Table, by using Create method with
    //single parameter.
    HmiTag hmiTag4 = hmiTags2.Create("Tag_4");
}
```

Deleting HMI tags

Modify the following program code to delete a HMI tag:

```
private void DeleteTag()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
//1. Delete tag at HmiSoftware level.
HmiTagComposition hmiTags1 = hmiSoftware.HmiTags;
HmiTag hmiTag1 = hmiTags1.Find("Tag1");
hmiTag1.Delete();
//2. Delete tag at tag table level.
HmiTagComposition hmiTags2 = hmiSoftware.HmiTagTables.Find("Default tag table").HmiTags;
HmiTag hmiTag2 = hmiTags2.Find("Tag2");
hmiTag2.Delete();
}
```

Enumerating HMI tags

Modify the following program code to enumerate HMI tag:

```
private void BrowseTag()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
//1. User can navigate all tag of device with following code.
HmiTagComposition hmiTags = hmiSoftware.HmiTags;
foreach (HmiTag hmiTag in hmiTags)
{
//...
}
//2. User can navigate all tags of given tag table with following code.
HmiTagComposition hmiTags2 = hmiSoftware.HmiTagTables.Find("Default tag table").HmiTags;
foreach (HmiTag hmiTag2 in hmiTags2)
{
//...
}
```

Accessing a HMI tag

Modify the following program to access single HMI tag by name:

```
private void AccessTag()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
//User can search tag present in UA device with following code.
HmiTag hmiTag3 = hmiSoftware.HmiTags.Find("Tag1");
}
```

Modify the following program code to access a HMI tag by table name:

```
{
HmiSoftware hmiSoftware = GetHmiSoftware();
// User can search tag present in given tag table with following code.
HmiTag hmiTag4 = hmiSoftware.HmiTagTables.Find("Default tag
table").HmiTags.Find("Tag1");
}
```

Accessing HMI tag properties

Modify the following program code to accessing properties of HMI tag:

```
private void TagProperties()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
//Tag_1 exists in TIA device project.
HmiTag hmiTag = hmiSoftware.HmiTags.Find("Tag_1");
//Enumeration type properties
AccessMode accessMode = hmiTag.AccessMode;
UpdateScope scope = hmiTag.UpdateScope;
//Substitute value depends on data type of tag.
hmiTag.SubstituteValue.Value = "144";
hmiTag.SubstituteValue.OnCommunicationError = false;
hmiTag.Name = "Tag_2";
//Assign valid cycle name.
hmiTag.AcquisitionCycle = "T1s";
hmiTag.DataType = "int";
hmiTag.PlcTag = "PlcTag_1";
//Start value depends on data type of tag.
hmiTag.StartValue = "1";
hmiTag.LowerLimit.ValueType = LimitValueType.Constant;
hmiTag.LowerLimit.Value = "HmiTag_1";
//Comment property (multilingual text)
string culture = "en-US";
//get the language based on given culture. Culture is standard and it should have correct
value.
Language language = GetDeviceProject().LanguageSettings.Languages.Find(new
System.Globalization.CultureInfo(culture));
//get comment as multilingual text.
MultilingualText multiLingualComment = hmiTag.Comment;
//get all text items from comment property.
MultilingualTextItemComposition texts = multiLingualComment.Items;
//find text from text list based on language.
MultilingualTextItem textItemEnglish = texts.Find(language);
//get value in selected culture.
string commentEng = textItemEnglish.Text;
}
```

Usage of UDT datatype for tags

You can modify and use the following program code to assign UDT datatype for tags and access member tags.

Assigning UDT datatype to HMI tag

Modify the following program code to assign UDT datatype to HMI tag with no connection:

```
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiTagComposition tags = hmiSoftware.HmiTags;
tags[0].Data Type = @"\Project library\Types\New folder_3\HmiUdt_string\V
0.0.1";
```

Modify the following program code to assign UDT datatype to HMI tag with connection:

```
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiTagComposition tags = hmiSoftware.HmiTags;
tags[0].Connection = "Connection_1";
tags[0].Data Type = @"\Project library\Types\New folder_3\HmiUdt_string\V
0.0.1";
```

Note

You can use full paths to assign Hmi UDT as data type

Modify the following program code to assign PLC UDT to HMI tag with connection:

```
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiTagComposition tags = hmiSoftware.HmiTags;
tags[0].Connection = "HMI_Connection_1";
tags[0].Data Type = "PlcTag_1";
```

Accessing datatype member properties for HMI tag

Modify the following program code to access datatype member properties for HMI tag:

```
private static void GetProperties(HmiTagComposition hmiTags)
{
foreach (HmiTag hmiTag in hmiTags)
{
switch(hmiTag.TagType)
{
case HmiTagType.Simple
Console.WriteLine("Name : "+ hmiTag.Name);
GetSimpleTagProperties(HmiTag hmiTag)
break;
case HmiTagType.UDT;
case HmiTagType.Array;
GetProperties(hmiTag.Members); //recursive call to current function.
break;
}
}
}

private static void GetSimpleTagProperties(HmiTag hmiTag)
{
Console.WriteLine("Name : " +hmiTag.Name);
Console.WriteLine("Date type : " +hmiTag.DataType);
// List the properties of hmi tag
//
//...
}

private static void GetProperties(HmiTag hmiTag.Members)
{
Console.Writeline("Name : " +hmiTag.Members.Name);
Console.Writeline("Date type: " +hmiTag.Members.DataType);
// List the properties of hmi tag members
//...
//...
}
```

Properties of member tags of UDT

You can modify and use the following program code to set and get member tags properties of user defined datatype.

7.18 Functions for accessing the data of an HMI device (Unified)

Modify the following program code to set properties of member tags of user defined datatype:

```

public HmiTag GetTag()
{
    DeviceItem deviceItem = Tiaproject.Devices[0].DeviceItems[1];
    SoftwareContainer softCont = deviceItem.GetService<SoftwareContainer>();
    HmiSoftware hmiSoftware = softCont.Software as HmiSoftware;
    if (hmiSoftware == null)
    {
        Device device = Tiaproject.Devices.Find("PC-System_1");
        DeviceItem deviceItem1 = device.DeviceItems[1];
        SoftwareContainer softCont1 = deviceItem1.GetService<SoftwareContainer>();
        hmiSoftware = softCont1.Software as HmiSoftware;
    }
    HmiTagComposition tagComposition = hmiSoftware.Tags;
    var tag = tagComposition.Create("Test_Tag");
    return tag;
}
public void ReadWrite_HMIUDTTagDisplayNamePropertyWithInternalConn_Succeeds()
{
    HmiTag tag = GetTag();
    tag.DataType = "HmiUdt_1_WithInternalConn V 0.0.1";
    for (int i = 0; i < tag.Members.Count; i++)
    {
        for (int j = 0; j < tag.Members[i].DisplayName.Items.Count; j++)
        {
            tag.Members[i].DisplayName.Items[j].Text = "DisplayNameTest" + j;
            Console.WriteLine("Display Name : ", tag.Members[i].DisplayName.Items[j].Text);
        }
    }
}
/// <summary>
/// HMI UDT Tag Display Name Property with S7 1500 PLC Connection.
/// </summary>
public void ReadWrite_HMIUDTTagDisplayNamePropertyWithS71500_Succeeds()
{
    HmiTag tag = GetTag();
    tag.Connection = "HMI_Connection_1";
    tag.DataType = "HmiUdt_1_WithS71500Conn V 0.0.1";
    for (int i = 0; i < tag.Members.Count; i++)
    {
        for (int j = 0; j < tag.Members[i].DisplayName.Items.Count; j++)
        {
            tag.Members[i].DisplayName.Items[j].Text = "DisplayNameTest" + j;
            Console.WriteLine("Display Name : ", tag.Members[i].DisplayName.Items[j].Text);
        }
    }
}

```

Modify the following program code to get access properties of member tags of user defined datatype:

```

public HmiTag GetTag()
{
DeviceItem deviceItem = Tiaproject.Devices[0].DeviceItems[1];
SoftwareContainer softCont = deviceItem.GetService<SoftwareContainer>();
HmiSoftware hmiSoftware = softCont.Software as HmiSoftware;
if (hmiSoftware == null)
{
Device device = Tiaproject.Devices.Find("PC-System_1");
DeviceItem deviceItem1 = device.DeviceItems[1];
SoftwareContainer softCont1 = deviceItem1.GetService<SoftwareContainer>();
hmiSoftware = softCont1.Software as HmiSoftware;
}
HmiTagComposition tagComposition = hmiSoftware.Tags;
var tag = tagComposition.Create("Test_Tag");
return tag;
}
public void Read_HMIUDTTagNamePropertyWithInternalConn_Succeeds()
{
HmiTag tag = GetTag();
tag.DataType = "HmiUdt_1_WithInternalConn V 0.0.1";
for (int i = 0; i < tag.Members.Count; i++)
{
var name = tag.Members[i].Name;
Console.WriteLine("Name : ", name );
}
}
/// <summary>
/// HMI UDT Tag Display Name Property with S71500 connection
/// </summary>
public void Read_HMIUDTTagNamePropertyWithS71500Conn_Succeeds()
{
HmiTag tag = GetTag();
tag.Connection = "HMI_Connection_1";
tag.DataType = "HmiUdt_1_WithS71500Conn V 0.0.1";
for (int i = 0; i < tag.Members.Count; i++)
{
var name = tag.Members[i].Name;
Console.WriteLine("Name : ", name );
}
}

```

See also

[HMI Unified Software object \(Page 272\)](#)

7.18.6.3 Working with system tags

Introduction

You can perform the following tasks with system tags while using TIA Portal Openness:

- Access a system tag
- Access system tag properties

Properties

The following properties are supported in System tag:

Property Name	Data Type	Description	Access
Name	String	Specify System Tag Name	R
DataType	String	Specify data type of System Tag	R

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76).
- A project is open.
See Opening a project (Page 106).
- Access to the HMI Software object
See HMI Unified Software object (Page 272)

Program code

Accessing a system tag

Modify the following program code to access single system tag by name:

```
private void SystemTagName()
{
    HmiSoftware hmiSoftware = GetHmiSoftware();
    HmiSystemTagComposition hmiSystemTags = hmiSoftware.HmiSystemTags;
    //Find system tag by using its name.
    HmiSystemTag hmiSystemTagObj = hmiSystemTags.Find("@UserName");
}
```

Accessing system tag properties

Modify the following program code to access properties of system tags:

```
private void SystemTagProperties()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiSystemTagComposition hmiSystemTags = hmiSoftware.HmiSystemTags;
HmiSystemTag hmiSystemTag = hmiSystemTags.Find("@UserName");
string name = hmiSystemTag.Name;
string dataType = hmiSystemTag.DataType;
}
```

See also

[HMI Unified Software object \(Page 272\)](#)

7.18.7 Connections

7.18.7.1 Working with connections

Introduction

You can perform the following tasks with connections while using TIA Portal Openness:

- Create connections
- Delete connections
- Enumerate connections
- Access connection properties

Properties

The following properties are supported in HMI Connections:

Property Name	Data Type	Description	Access
Name	String	Name of connection	R/W
DisabledAtStartup	Boolean	Connection initially will be online or not in runtime.	R/W
CommunicationDriver	String	Gives the communication driver	R/W
Node	String	Shows access point of Partner (eg PLC).	R
Partner	String	Name of connected PLC.	R

Property Name	Data Type	Description	Access
Station	String	Name of the station to which PLC is located.	R
Comment	String	Additional comments if any	R/W
InitialAddress	String	Provide parameters of connection like type of interface (DP, TCP/IP), IP address, Rack etc.	R/W

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
Connecting to the TIA Portal (Page 76)
- A project is open.
Opening a project (Page 106)

Program code

You can modify and use the following program code examples while working with connections.

Creating connections

Modify the following program code to create a connection:

```
private void ConnectionCreate()
{
//Create Connection
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiConnectionComposition connections = hmiSoftware.HmiConnections;
HmiConnection connection = connections.Create("Connection_1");
}
```

Deleting connections

Modify the following program code to delete a connection:

```
private void ConnectionDelete()
{
//User wants to delete connection with name "Connection_1"
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiConnectionComposition connections = hmiSoftware.HmiConnections;
HmiConnection connection = connections.Find("Connection_1")
connection.Delete();
}
```

Enumerating connections

Modify the following program code to enumerate connection:

```
private void ConnectionBrowse()
{
//Browse Connections
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiConnectionComposition connections = hmiSoftware.HmiConnections;
foreach (HmiConnection connection in connections)
{
//Work with connections
}
//Other way to get connection by using Find function on connection Composition
HmiConnection connectionObj = connections.Find("Connection_1");
}
```

Accessing connection properties

Modify the following program code to accessing connection properties:

```

private void ConnectionProperties()
{
//Set/Get Connection properties.
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiConnectionComposition connections = hmiSoftware.HmiConnections;
HmiConnection connection = connections.Find("Connection_1");
string name = connection.Name;
connection.Name = "Connection_2";
bool online = connection.DisabledAtStartup;
connection.DisabledAtStartup = true;
string node = connection.Node;
string station = connection.Station;
string partner = connection.Partner;
string communicationDriver = connection.CommunicationDriver;
connection.CommunicationDriver = "SIMATIC S7 300/400";
//valid Initial Address strings
connection.InitialAddress = "PlcRack = 4";
connection.InitialAddress = "HostAddress = 127.157.2.2";
connection.InitialAddress = "hostAddress = 127.157.2.2";//key name is case insensitive so
it is valid"
connection.InitialAddress = "CommunicationInterface = Industrial Ethernet; HostAddress =
127.157.8.1; HostAccessPoint = S7ONLINE; PlcExpansionSlot = 1; PlcRack = 1;
PlcIsCyclicOperation = false";
connection.InitialAddress = "CommunicationInterface = Industrial Ethernet; HostAddress =
127.157.8.1; HostAccessPoint = S7ONLE; PlcRack = 5; PlcIsCyclicOperation = true";
//Below are Wrong Value or Wrong Formats which leads to recoverable exceptions.
//Wrong Value
connection.InitialAddress = null;
connection.InitialAddress = string.Empty;
connection.InitialAddress = "HostAddress = 127.157.2.2.1";
connection.InitialAddress = "HostAddress = 127.157.2.2.1; PlcRack = 5";
//key value format is not followed.
connection.InitialAddress = "HostAccessPoint = S7ONINE; PlcAddress * 155.166.8.2;
PlcExpansionSlot ; 1; PlcRack = 5; PlcIsCyclicOperation = true";
//semicolon with empty key value pair
connection.InitialAddress = ";;CommunicationInterface = Industrial Ethernet; HostAddress =
127.157.0.1";
connection.InitialAddress = ";CommunicationInterface = Industrial Ethernet; HostAddress =
127.157.0.1";
connection.InitialAddress = "CommunicationInterface = Industrial Ethernet; HostAddress =
127.157.0.1;";
connection.InitialAddress = "CommunicationInterface = Industrial Ethernet;; HostAddress =
127.157.0.1;";
connection.CommunicationDriver = "SIMATIC S7 1500";
//wrong as PlcRack is not valid key / property for Simatic S71500 connection.
connection.InitialAddress = "PlcRack = 4";
}

```

7.18.8 Runtime settings

7.18.8.1 Working with runtime settings

Introduction

You can perform the following tasks with runtime settings while using TIA Portal Openness:

- Read/write runtime settings

Properties

The following properties are supported in Runtime Settings:

Property Name	Data Type	Description	Access
HmiSoftware.RuntimeSettings	HmiRuntimeSetting	Runtime Setting object	R
RuntimeSettings.StartScreen	String	Start Screen.	R/W
RuntimeSettings.OperateAsOpcServer	Boolean	Specifies that given device should act as OPC Server.	R/W
RuntimeSettings.LanguageAndFonts	HmiLanguageAndFont Association	List of language and fonts	R
RuntimeSettings.LanguageAndFonts [index].Enable	Boolean	Enable / Disable runtime language.	R/W
RuntimeSettings.LanguageAndFonts [index].EnableForLogging	Boolean	Specifies language use for logging on runtime	R/W
RuntimeSettings.LanguageAndFonts [index].Order	Short	Specifies order for transferring fonts to device.	R
RuntimeSettings.LanguageAndFonts [index].Language	String	Name of language	R
RuntimeSettings.LanguageAndFonts [index].Fixed-Font1	String	Predefined font family available for font configuration	R
RuntimeSettings.LanguageAndFonts [index].Fixed-Font2	String	Predefined font family available for font configuration	R
RuntimeSettings.LanguageAndFonts [index].Fixed-Font3	String	Predefined font family available for font configuration	R
RuntimeSettings.LanguageAndFonts [index].Fixed-Font4	String	Predefined font family available for font configuration	R

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- Access to the HMI Software object.
See HMI Unified Software object (Page 272)

Program code

```
private void RuntimeSettingsPropertiesAccess()
{
    HmiSoftware hmiSoftware = GetHmiSoftware();
    hmiSoftware.RuntimeSettings.OperateAsOpcServer = true;
    hmiSoftware.RuntimeSettings.StartScreen = "Screen_1";
    hmiSoftware.RuntimeSettings.LanguageAndFonts[0].Enable = true;
    hmiSoftware.RuntimeSettings.LanguageAndFonts[0].EnableForLogging = true;
}
```

7.18.9 Screens and dynamizations

7.18.9.1 Working with screens

Introduction

You can perform the following tasks with screens while using TIA Portal Openness:

- Creating screens
- Deleting screens
- Enumerating screens
- Accessing screen properties

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to the HMI Software object
See HMI Unified Software object (Page 272)

Program code

You can modify and use the following program code example while working with screens.

Creating screens

Modify the following program code to create screens:

```
HmiSoftware hmisoftware = GetHMIsoftware();
HmiScreenComposition objScreens = hmisoftware.Screens;
HmiScreen objHmiScreen = objScreens.Create("TestScreen");
```

Recoverable Exception will be raised if the Create method is called with Screen Name with which Screen already exists in TIA Portal.

Deleting screens

Modify the following program code to delete screens:

```
public void Delete()
//Case 1
HmiSoftware hmisoftware = GetHMIsoftware();
HmiScreenComposition objScreens = hmisoftware.Screens;
HmiScreen objHmiScreen = objScreens.Create("TestScreen");
if (objHmiScreen != null)
{
objHmiScreen.Delete();
}
//Case 2
HmiSoftware hmisoftware = GetHMIsoftware();
HmiScreenComposition objScreens = hmisoftware.Screens;
objScreens.Create("TestScreen1");
IEngineeringObject alarmClassEnggObj = (IEngineeringObject) objScreens[0];
if (alarmClassEnggObj != null)
{
alarmClassEnggObj.Invoke("Delete", null);
}
```

Enumerating screens

Modify the following program code to enumerate all screens of device:

```
HmiSoftware hmiSoftware = GetHMIsoftware();
HmiScreenComposition objScreens = hmiSoftware.Screens;
foreach (HmiScreen ObjHmiScreen in objScreens)
{
//work with Screens
}
```

Modify the following program code to searching screen from screens list on the basis of name:

```
HmiScreen objHmiScreen = objScreens.Find("TestScreen3");
```

7.18 Functions for accessing the data of an HMI device (Unified)

Modify the following program code to get screen from screens list on basis of index:

```
HmiScreen objHmiScreen = hmisoftware.Screens[0];
```

Modify the following program code to checking a particular screen item exist in screen item list by using Contains method:

```
HmiScreenComposition objScreens = hmisoftware.Screens;
HmiScreen screen1 = objScreens.Create("TestScreen1");
bool isExist = objScreens.Contains(screen1);
```

Accessing screen properties

Modify the following program code to set and get properties of screen:

```
HmiSoftware hmisoftware = GetHmiSoftware();
HmiScreenComposition objScreens = hmisoftware.Screens;
HmiScreen objHmiScreen = objScreens.Create("TestScreen");
objHmiScreen.Width = 50;
objHmiScreen.ScreenNumber = 1;
objHmiScreen.BackGraphic = "DownArrow";
objHmiScreen.DisplayName.Items[0].Text = "screen";
objHmiScreen.Name = "ScreenTest";
objHmiScreen.Enabled = true;
objHmiScreen.Height = 123;
```

Modify and use the following program code to get all properties of screen using GetAttributes():

```
HmiSoftware hmisoftware = GetHMISoftware();
HmiScreenComposition objScreens = hmisoftware.Screens;
HmiScreen objHmiScreen = objScreens.Create("TestScreen");
List<string> getlststring = new List<string>();
{
"AlternateBackColor", "BackColor", "BackFillPattern", "BackGraphic",
"BackGraphicStretchMode", "BackgroundFillMode", "DisplayName", "Dynamization", "Enabled",
"EnabledExplicitRelease", "Height", "HorizontalAlignment", "Layers", "Name", "Parent",
"ScreenElements", "ScreenItems", "ScreenNumber", "VerticalAlignment", "Width"
};
var getallpropValue = objHmiScreen.GetAttributes(getlststring);
```

Modify the following program code to set all properties of screen using SetAttributes():

```
HmiSoftware hmisoftware = GetHMISoftware();
HmiScreenComposition objScreens = hmisoftware.Screens;
HmiScreen objHmiScreen = objScreens.Create("TestScreen");
Dictionary<string, object> setPropertyName = new Dictionary<string, object>();
setPropertyName.Add("AlternateBackColor", Color.Aqua);
setPropertyName.Add("BackColor", Color.Aqua);
setPropertyName.Add("BackFillPattern", HmiFillPattern.GradientHorizontalTricolor);
setPropertyName.Add("BackGraphic", "DownArrow");
setPropertyName.Add("BackGraphicStretchMode", HmiGraphicStretchMode.Fill);
setPropertyName.Add("BackgroundFillMode", HmiBackgroundFillMode.Screen);
setPropertyName.Add("DisplayName", "AnalogAlarmByDiffMethod");
setPropertyName.Add("Enabled", "AnalogAlarmByDiffMethod");
setPropertyName.Add("EnabledExplicitRelease", "AnalogAlarmByDiffMethod");
setPropertyName.Add("Height", "AnalogAlarmByDiffMethod");
setPropertyName.Add("HorizontalAlignment", "AnalogAlarmByDiffMethod");
setPropertyName.Add("Name", "AnalogAlarmByDiffMethod");
setPropertyName.Add("ScreenNumber", "AnalogAlarmByDiffMethod");
setPropertyName.Add("VerticalAlignment", "AnalogAlarmByDiffMethod");
setPropertyName.Add("Name", "AnalogAlarmByDiffMethod");
setPropertyName.Add("Width", "AnalogAlarmByDiffMethod");
objHmiScreen.SetAttributes(setPropertyName);
```

7.18 Functions for accessing the data of an HMI device (Unified)

Modify and use the following program code to get the value from property using IEngineeringObject's GetAttribute:

```
HmiSoftware hmisoftware = GetHMISoftware();
HmiScreenComposition objScreens = hmisoftware.Screens;
HmiScreen objHmiScreen = objScreens.Create("Testscreen");
IEngineeringObject obj = objHmiScreen;
Color clraltnvbk = (Color)obj.GetAttribute("AlternateBackColor");
Color clrbk = (Color)obj.GetAttribute("BackColor");
HmiFillPattern filptrns = (HmiFillPattern)obj.GetAttribute("BackFillPattern");
string bkgrphic = (string)obj.GetAttribute("BackGraphic");
HmiGraphicStretchMode strmode =
(HmiGraphicStretchMode)obj.Getattribute("BackGraphicStretchmode");
HmiBackgroundFillMode bkgrndfilmode = (HmiBackground
dFillMode)obj.GetAttribute("BackgroundFillMode");
string dsplyName = (string)obj.GetAttribute("DisplayName");
DynamizationBaseComposition dymztns =
(DynamizationBaseComposition)obj.GetAttribute("Dynamizations");
b001 enble = (b001)obj.GetAttribute("Enabled");
SomRef objsom = (SomRef)obj.GetAttribute("EnableExplicitRelease");
uint fight = (uint)obj.GetAttribute("Height");
HmiHorizontalAlignment hrzntl =
(HmiHorizontalAlignment)obj.GetAttribute("HorizontalAlignment");
HmiverticalAlignment vrtcle = (HmiverticalAlignment)obj.GetAttribute("VerticalAlignment");
HmiLayerPartComposition objlyrs = (HmiLayerPartComposition)Obj.GetAttribute("layers");
string name = (string)obj.GetAttribute("Name");
IEngineeringObject objparen = (IEngineeringobject)obj.GetAttribute("Parent");
HmiscreenElementBaseComposition objelemnt =
(HmiscreenElementBaseComposition)obj.GetAttribute("ScreenElements");
HmiScreenItemBaseComposition objitems =
(HmiScreenItemBaseComposition)obj.GetAttribute("ScreenItems");
byte scrnnumber = (byte)obj.GetAttribute("ScreenNumber");
uint width = (uint)obj.GetAttribute(Width");
```

Modify the following program code to determine if all properties have valid values:

```
HmiSoftware hmiSoftware = GetHMIsoftware ();
HmiScreen objHmiScreen = hmiSoftware.Screens.Create("HmiScreen_1");
objHmiScreen.DisplayName.Items[0].Text = GetMorethan512BoundaryString();
IList<IValidator> objectsToValidate = new List<IValidator>();
foreach (var item in hmiSoftware.Screens)
{
    objectsToValidate.Add(item);
}
foreach (IValidator validator in objectsToValidate)
{
    IList<HmiValidationResult> errors = validator.Validate();
    if (errors != null && errors.Count > 0)
    {
        foreach (var errornotification in errors)
        {
            var propName = errornotification.PropertyName;
            foreach (var errormassage in errornotification.Errors)
            {
                Console.WriteLine(errormassage);
            }
        }
    }
}
```

Note

If during set of properties the set operation is not able to set the value then a Recoverable exception would be raised.

7.18.9.2 Basic screen items

Working with basic screen items

Introduction

You can perform the following tasks with basic screen items while using TIA Portal Openness:

- Creating screen items (Basic)
- Deleting screen items (Basic)
- Enumerating screen items (Basic)

The following basic screen items are supported by the TIA Portal Openness:

Table 7-1 Types and Namespace (s)

Screen Item	Class	Namespace (s)
Line	HmiLine	Siemens.Engineering.HmiUnified.UI.Shapes Siemens.Engineering.HmiUnified.UI.Base
Polyline	HmiPolyline	
Polygon	HmiPolygon	
Ellipse	HmiEllipse	
Ellipse Segment	HmiEllipseSegment	
Circle Segment	HmiCircleSegment	
Elliptical arc	HmiEllipticalArc	
Circular arc	HmiCircularArc	
Circle	HmiCircle	
Rectangle	HmiRectangle	
Graphic view	HmiGraphicView	

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 76)

Program code

You can modify and use the following program code example while working with basic screen items.

Creating screen items (Basic)

Modify the following program code to create basic screen items:

```
HmiSoftware hmiSoftware = GetHMISoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
HmiLine hmiLine = screenitems.Create<HmiLine>("ScreenItem_1");
```

Note

The above example is applicable to all the screen items defined in type table above. You need to change the HmiLine with type(s) defined in Types and Namespace(s) table.

Recoverable Exception will be raised if the Create method is called with Screen item Name with which Screen item already exists in TIA.

Deleting screen items (Basic)

Modify the following program code to delete basic screen items.

```
public void Delete()
//Case 1
HmiSoftware hmiSoftware = GetHMISoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screen[0].ScreenItems;
HmiLine hmiLine = screenitems.Create<HmiLine>("ScreenItem_1");
if (hmiLine != null)
{
hmiLine.Delete();
}
//Case 2
HmiSoftware hmiSoftware = GetHMISoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Scren1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
HmiLine hmiLine = screenitems.Create<HmiLine>("ScreenItem_1");
IEngineeringObject ObjhmiLineEnggObj = hmiLine;
if (ObjhmiLineEnggObj != null)
{
ObjHmiScreenItemssEnggObj.Invoke("Delete", null);
}
```

Note

The above example is applicable to all the screen items defined in type table above. You need to change the HmiLine with type(s) defined in Types and Namespace(s) table.

Enumerating screen items (Basic)

Modify the following program code to enumerate all screen items of screen.

```
HmiSoftware hmiSoftware = GetHMISoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
foreach (var item in screenitems)
{
//work with screenitems
}
```

Modify the following program code to find Screen item from Screen items list on the basis of name:

```
HmiScreenItemBase screenitems = hmiSoftware.Screens[0].ScreenItems.Find("ScreenItems_1");
```

7.18 Functions for accessing the data of an HMI device (Unified)

Modify the following program code to get Screen item from ScreenItems list on basis of index:

```
HmiScreenItemBase screenitem = hmiSoftware.Screens[0].ScreenItems[0];
```

Modify the following program code to check a particular screen item exist in Screen item list by using Contains method:

```
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
HmiLine hmiLine = screenitems.Create<HmiLine>("ScreenItems_1");
bool isexists = screenitems.Contains(hmiLine);
```

Note

The above example is applicable to all screen items defined in type table. You need to change the HmiLine with type(s) defined in Types and Namespace(s) table.

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

[HMI Unified Software object \(Page 272\)](#)

Accessing properties of line

Introduction

You can use the TIA Portal Openness to get and set properties of line screen item.

The following line properties are supported in screen item:

Property Name	Property Type	Accessibility
X1	int	False
Y1	int	False
X2	int	False
Y2	int	False
Enabled	Bool	True
CurrentQuality	HmiQuality	False
LineColor	Color	False
AlternateLineColor	Color	False
DashType	HmiDashType	False
EndType	HmiLineEndType	False
StartType	HmiLineStartType	False
CapType	HmiCapType	False
LineWidth	byte	False

Property Name	Property Type	Accessibility
Top	int	False
Left	int	False
Width	uint	False
Height	uint	False
RotationAngle	short	False
RotationCenterX	float	False
RotationCenterY	float	False
RotationCenterPlacement	HmiRotationCenterPlacement	False
Opacity	float	False
Name	string	False
Visible	bool	False
TabIndex	ushort	False
ToolTipText	MultilingualText	False
Authorization	string	False

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with basic screen items (Page 317)

Program code

Modify the following code to access basic properties of Line:

```
//Name
var name = hmiline.Name;
hmiline.Name = "Default Value";
//AlternateLineColor
var linecolor = hmiline.AlternateLineColor;
hmiline.AlternateLineColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//Height
var height = hmiline.Height;
hmiline.Height = 100;
```

Modify the following code to access multilingual property:

```
//ToolTipText
var tooltip = hmiline.ToolTipText;
var tooltiptext = hmiline.ToolTipText.Items[0].Text;
hmiline.ToolTipText.Items[0].Text = "<body><p>TestforMultilinugualProperty</p></body>";
```

Modify the following code to access other typical property of Line:

```
//DashType
var dashtype = hmiline.DashType;
hmiline.DashType = HmiDashType.DashDotDot;
```

See also

- [Connecting to the TIA Portal \(Page 76\)](#)
- [Opening a project \(Page 106\)](#)
- [Working with basic screen items \(Page 317\)](#)
- [HMI Unified Software object \(Page 272\)](#)

Accessing properties of polyline

Introduction

You can use the TIA Portal Openness to get and set properties of polyline.

The following polyline properties are supported in basic screen item:

Property Name	Property Type	Accessibility
LineColor	Color	False
Enabled	Bool	False
CurrentQuality	HmiQuality	True
RotationCenterPlacement	HmiRotationCenterPlacement	False
AlternateLineColor	Color	False
DashType	HmiDashType	False
EndType	HmiLineEndType	False
StartType	HmiLineStartType	False
CapType	HmiCapType	False
LineWidth	byte	False
JoinType	HmiLineJoinType	False
Top	int	False
Left	int	False
Width	uint	False
Height	uint	False

Property Name	Property Type	Accessibility
RotationAngle	short	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
Name	string	False
Visible	bool	False
TabIndex	ushort	False
ToolTipText	MultilingualText	False
Authorization	string	False

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with basic screen items (Page 317)

Program code

Modify the following program code to access basic properties of Polyline:

```
//Name
var name = polyline.Name;
polyline.Name = "Default Value";
//AlternateLineColor
var linecolor = polyline.AlternateLineColor;
polyline.AlternateLineColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//Height
var height = polyline.Height;
polyline.Height = 100;
```

Modify the following program code to access multilingual property:

```
//ToolTipText
var tooltip = polyline.ToolTipText;
var tooltiptext = polyline.ToolTipText.Items[0].Text;
polyline.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Modify the following program code to access other typical property:

```
//Joint Type
var dashtype = polyline.JoinType;
polyline.JoinType = HmiLineJoinType.Miter;
//Points
var points = polyline.Points;
var point = points[0];
var x = point.X;
point.X = 10;
var y = point.Y;
point.Y = 10;
var newPoint = points.Create(15, 100);
```

See also

- [Connecting to the TIA Portal \(Page 76\)](#)
- [Opening a project \(Page 106\)](#)
- [HMI Unified Software object \(Page 272\)](#)
- [Working with basic screen items \(Page 317\)](#)

Accessing properties of ellipse

Introduction

You can use the TIA Portal Openness to get/set properties of ellipse screen item.

The following ellipse properties are supported in basic screen item:

Property Name	Property Type	Accessibility
BorderColor	Color	False
AlternateBorderColor	Color	False
BackColor	Color	False
AlternateBackColor	Color	False
BorderWidth	byte	False
Enabled	Bool	False
CurrentQuality	HmiQuality	True
RotationCenterPlacement	HmiRotationCenterPlacement	False
BackFillPattern	HmiFillPattern	False
FillLevel	byte	False
FillDirection	HmiFillDirection	False
DashType	HmiDashType	False
ShowFillLevel	bool	False
RadiusX	uint	False
RadiusY	uint	False
CenterX	int	False

Property Name	Property Type	Accessibility
CenterY	int	False
RotationAngle	short	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
Name	string	False
Visible	bool	False
TabIndex	ushort	False
ToolTipText	MultilingualText	False
Authorization	string	False

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- Opening a Project
See Opening a Project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with basic screen items (Page 317)

Program code

Modify the following program code to access basic properties of Ellipse:

```
//Name
var name = ellipse.Name;
ellipse.Name = "Default Value";
//AlternateBorderColor
var bordercolor = ellipse.AlternateBorderColor;
ellipse.AlternateBorderColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//BorderWidth
var borderwidth = ellipse.BorderWidth;
ellipse.BorderWidth = 10;
```

Modify the following program code to access Multilingual property:

```
//ToolTipText
var tooltip = ellipse.ToolTipText;
var tooltiptext = ellipse.ToolTipText.Items[0].Text;
ellipse.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Modify the following program code to access other typical property of ellipse:

```
//BackFillPattern
var backfill = ellipse.BackFillPattern;
ellipse.BackFillPattern = HmiFillPattern.GradientBackwardDiagonal;
```

See also

- [Connecting to the TIA Portal \(Page 76\)](#)
- [Opening a project \(Page 106\)](#)
- [HMI Unified Software object \(Page 272\)](#)
- [Working with basic screen items \(Page 317\)](#)

Accessing properties of ellipse segment

Introduction

You can use the TIA Portal Openness to get and set properties of ellipse segment.

The following ellipse segment properties are supported in screen item:

Property Name	Property Type	Accessibility
StartAngle	int	False
AngleRange	int	False
BorderColor	Color	False
AlternateBorderColor	Color	False
BackColor	Color	False
CurrentQuality	HmiQuality	True
RotationCenterPlacement	HmiRotationCenterPlacement	False
BorderWidth	byte	False
BackFillPattern	HmiFillPattern	False
FillLevel	byte	False
FillDirection	HmiFillDirection	False
DashType	HmiDashType	False
ShowFillLevel	bool	False
RadiusX	uint	False
RadiusY	uint	False
CenterX	int	False
CenterY	int	False
RotationAngle	short	False
RotationCenterX	float	False
RotationCenterY	float	False
RotationCenterPlacement	HmiRotationCenterPlacement	False
Opacity	float	False

Property Name	Property Type	Accessibility
Name	string	False
Visible	bool	False
TabIndex	ushort	False
ToolTipText	MultilingualText	False
Authorization	string	False

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- Opening a Project
See Opening a Project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with basic screen items (Page 317)

Program code

Modify the following program code to access basic properties of Ellipse Segment:

```
//Name
var name = ellipsesegment.Name;
ellipsesegment.Name = "Default Value";
//AlternateBorderColor
var bordercolor = ellipsesegment.AlternateBorderColor;
ellipsesegment.AlternateBorderColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//BorderWidth
var borderwidth = ellipsesegment.Height;
ellipsesegment.BorderWidth = 10;
```

Modify the following program code to access multilingual property:

```
//ToolTipText
var tooltip = ellipsesegment.ToolTipText;
var tooltiptext = ellipsesegment.ToolTipText.Items[0].Text;
ellipsesegment.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Modify the following program code to access other typical property:

```
//BackFillPattern
var backfill = ellipsesegment.BackFillPattern;
ellipsesegment.BackFillPattern = HmiFillPattern.GradientBackwardDiagonal;
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

[HMI Unified Software object \(Page 272\)](#)

Accessing properties of circle segment

Introduction

You can use the TIA Portal Openness to get and set properties of circle segment.

The following circle segment properties are supported in screen item:

Property Name	Property Type	Accessibility
StartAngle	int	False
AngleRange	int	False
BorderColor	Color	False
AlternateBorderColor	Color	False
BackColor	Color	False
AlternateBackColor	Color	False
Enabled	Bool	False
CurrentQuality	HmiQuality	True
RotationCenterPlacement	HmiRotationCenterPlacement	False
BorderWidth	byte	False
BackFillPattern	HmiFillPattern	False
FillLevel	byte	False
FillDirection	HmiFillDirection	False
DashType	HmiDashType	False
ShowFillLevel	bool	False
Radius	uint	False
CenterX	int	False
CenterY	int	False
RotationCenterX	float	False
RotationCenterY	float	False
RotationAngle	short	False
Opacity	float	False
Name	string	False
Visible	bool	False
TabIndex	ushort	False
ToolTipText	MultilingualText	False
Authorization	string	False

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- Opening a Project
See Opening a Project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with basic screen items (Page 317)

Program

Modify the following program code to access basic property of Circle Segment:

```
//Name
var name = circlesegment.Name;
circlesegment.Name = "Default Value";
//AlternateBorderColor
var bordercolor = circlesegment.AlternateBorderColor;
circlesegment.AlternateBorderColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//BorderWidth
var borderwidth = circlesegment.BorderWidth;
circlesegment.BorderWidth = 10;
```

Modify the following program code to access multilingual property:

```
//ToolTipText
var tooltip = circlesegment.ToolTipText;
var tooltiptext = circlesegment.ToolTipText.Items[0].Text;
circlesegment.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Modify the following program code to access other typical property:

```
//ToolTipText
var tooltip = circlesegment.ToolTipText;
var tooltiptext = circlesegment.ToolTipText.Items[0].Text;
circlesegment.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

[HMI Unified Software object \(Page 272\)](#)

[Working with basic screen items \(Page 317\)](#)

Accessing properties of elliptical arc

Introduction

You can use the TIA Portal Openness to get and set properties of elliptical arc.

The following elliptical arc properties are supported in screen item:

Property Name	Property Type	Accessibility
StartAngle	int	False
AngleRange	int	False
LineColor	Color	False
AlternateLineColor	Color	False
StartType	HmiLineStartType	False
EndType	HmiLineEndType	False
CapType	HmiCapType	False
Enabled	Bool	False
CurrentQuality	HmiQuality	True
RotationCenterPlacement	HmiRotationCenterPlacement	False
DashType	HmiDashType	False
LineWidth	byte	False
RadiusX	uint	False
RadiusY	uint	False
CenterX	int	False
CenterY	int	False
RotationAngle	short	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
Name	string	False
Visible	bool	False
TabIndex	ushort	False
ToolTipText	MultilingualText	False
Authorization	string	False

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with basic screen items (Page 317)

Program code

Modify the following program code to access basic property of Elliptical arc:

```
//Name
var name = ellipticalarc.Name;
ellipticalarc.Name = "Default Value";
//LineColor
var linecolor = ellipticalarc.LineColor;
ellipticalarc.LineColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//LineWidth
var linewidth = ellipticalarc.LineWidth;
ellipticalarc.LineWidth = 10;
```

Modify the following program code to access multilingual property:

```
//ToolTipText
var tooltip = ellipticalarc.ToolTipText;
var tooltiptext = ellipticalarc.ToolTipText.Items[0].Text;
ellipticalarc.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Modify the following program code to access other typical property:

```
//CapType
var captype = ellipticalarc.CapType;
ellipticalarc.CapType = HmiCapType.Round;
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

[HMI Unified Software object \(Page 272\)](#)

[Working with basic screen items \(Page 317\)](#)

Accessing properties of circular arc

Introduction

You can use the TIA Portal Openness to get and set properties of circular arc.

The following circular arc properties are supported in screenitem:

Property Name	Property Type	Accessibility
StartAngle	int	False
AngleRange	int	False
LineColor	Color	False
AlternateLineColor	Color	False

7.18 Functions for accessing the data of an HMI device (Unified)

Property Name	Property Type	Accessibility
DashType	HmiDashType	False
EndType	HmiLineEndType	False
StartType	HmiLineEndType	False
Enabled	Bool	False
CurrentQuality	HmiQuality	True
RotationCenterPlacement	HmiRotationCenterPlacement	False
CapType	HmiCapType	False
LineWidth	byte	False
Radius	uint	False
CenterX	int	False
CenterY	int	False
RotationAngle	short	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
Name	string	False
Visible	bool	False
TabIndex	ushort	False
ToolTipText	MultilingualText	False
Authorization	string	False

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with basic screen items (Page 317)

Program code

Modify the following program code to access basic property of Circle arc:

```
//Name
var name = circulararc.Name;
circulararc.Name = "Default Value";
//LineColor
var linecolor = circulararc.LineColor;
circulararc.LineColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//LineWidth
var linewidth = circulararc.LineWidth;
circulararc.LineWidth = 10;
```

Modify the following program code to access multilingual property:

```
//ToolTipText
var tooltip = circulararc.ToolTipText;
var tooltiptext = circulararc.ToolTipText.Items[0].Text;
circulararc.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Modify the following program code to access other typical property :

```
//CapType
var captype = circulararc.CapType;
circulararc.CapType = HmiCapType.Round;
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

[HMI Unified Software object \(Page 272\)](#)

[Working with basic screen items \(Page 317\)](#)

Accessing properties of circle

Introduction

You can use the TIA Portal Openness to get and set properties of circle.

The following circle properties are supported in screenitem:

Property Name	Property Type	Accessibility
BorderColor	Color	False
AlternateBorderColor	Color	False
BackColor	Color	False
AlternateBackColor	Color	False

7.18 Functions for accessing the data of an HMI device (Unified)

Property Name	Property Type	Accessibility
Enabled	Bool	False
CurrentQuality	HmiQuality	True
RotationCenterPlacement	HmiRotationCenterPlacement	False
BorderWidth	byte	False
BackFillPattern	HmiFillPattern	False
FillLevel	byte	False
FillDirection	HmiFillDirection	False
DashType	HmiDashType	False
ShowFillLevel	bool	False
Radius	uint	False
CenterX	int	False
CenterY	int	False
RotationAngle	short	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
Name	string	False
Visible	bool	False
TabIndex	ushort	False
ToolTipText	MultilingualText	False
Authorization	string	False

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with basic screen items (Page 317)

Program code

Modify the following program code to access basic property of circle:

```
//Name
var name = circle.Name;
circle.Name = "Default Value";
//BorderColor
var bordercolor = circle.BorderColor;
circle.BorderColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//BorderWidth
var borderwidth = circle.BorderWidth;
circle.BorderWidth = 10;
```

Modify the following program code to access multilingual property:

```
//ToolTipText
var tooltip = circle.ToolTipText;
var tooltiptext = circle.ToolTipText.Items[0].Text;
circle.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Modify the following program code to access other typical property :

```
//DashType
var dashtype = circle.DashType;
circle.DashType = HmiDashType.Solid;
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

[HMI Unified Software object \(Page 272\)](#)

[Working with basic screen items \(Page 317\)](#)

Accessing properties of rectangle

Introduction

You can use the TIA Portal Openness to set and get properties of rectangle.

The following rectangle properties are supported in screen item:

Property Name	Property Type	Accessibility
BorderColor	Color	False
AlternateBorderColor	Color	False
BackColor	Color	False
AlternateBackColor	Color	False

7.18 Functions for accessing the data of an HMI device (Unified)

Property Name	Property Type	Accessibility
BorderWidth	byte	False
BackFillPattern	HmiFillPattern	False
Enabled	Bool	False
CurrentQuality	HmiQuality	True
RotationCenterPlacement	HmiRotationCenterPlacement	False
Corners	HmiCornersPart	True
FillLevel	byte	False
FillDirection	HmiFillDirection	False
DashType	HmiDashType	False
ShowFillLevel	bool	False
Top	int	False
Left	int	False
Width	uint	False
Height	uint	False
RotationAngle	short	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
Name	string	False
Visible	bool	False
TabIndex	ushort	False
ToolTipText	MultilingualText	False
Authorization	string	False

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with basic screen items (Page 317)

Program code

Modify the following program code to access basic properties:

```
//Name  
var name = rectangle.Name;  
rectangle.Name = "Default Value";  
//BorderColor  
var bordercolor = rectangle.BorderColor;  
rectangle.BorderColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);  
//BorderWidth  
var borderwidth = rectangle.BorderWidth;  
rectangle.BorderWidth = 10;
```

Modify the following program code to access multilingual property:

```
//ToolTipText  
var tooltip = rectangle.ToolTipText;  
var tooltiptext = rectangle.ToolTipText.Items[0].Text;  
rectangle.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Modify the following program code to access other typical property:

```
//Corners  
var corner = rectangle.Corners;  
var bottomleftradius = corner.BottomLeftRadius;  
corner.BottomLeftRadius = 50;  
var bottomrightradius = corner.BottomRightRadius;  
corner.BottomRightRadius = 30;  
var topleftradius = corner.TopLeftRadius;  
corner.TopLeftRadius = 50;  
var toprradius = corner.TopRightRadius;  
corner.TopRightRadius = 30;
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

[HMI Unified Software object \(Page 272\)](#)

[Working with basic screen items \(Page 317\)](#)

Accessing properties of graphic view

Introduction

You can use the TIA Portal Openness to set and get properties of graphic view.

The following rectangle properties are supported in screen item:

Property Name	Property Type	Accessibility
GraphicStretchMode	HmiGraphicStretchMode	False
FillLevel	byte	False
FillDirection	HmiFillDirection	False
ShowFillLevel	bool	False
BackFillPattern	HmiFillPattern	False
AlternateBackColor	Color	False
Enabled	Bool	False
CurrentQuality	HmiQuality	True
RotationCenterPlacement	HmiRotationCenterPlacement	False
Top	int	False
Left	int	False
Width	uint	False
Height	uint	False
Padding	HmiPaddingPart	True
RotationAngle	short	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
Name	string	False
Visible	bool	False
Graphic	string	False
TabIndex	ushort	False
ToolTipText	MultilingualText	False
Authorization	string	False

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with basic screen items (Page 317)

Program code

Modify the following program code to access basic property:

```
//Name  
var name = graphicview.Name;  
graphicview.Name = "Default Value";  
//AlternateBackColor  
var backcolor = graphicview.AlternateBackColor;  
graphicview.AlternateBackColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);  
//Height  
var height = graphicview.height;  
graphicview.Height = 10;
```

Modify the following program code to access multilingual property:

```
//ToolTipText  
var tooltip = graphicview.ToolTipText;  
var tooltiptext = graphicview.ToolTipText.Items[0].Text;  
graphicview.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Modify the following program code to access other typical property:

```
//Padding  
var padding = graphicview.Padding;  
var bottom = padding.Bottom;  
padding.Bottom = 50;  
var left = padding.Left;  
padding.Left = 60;  
var right = padding.Right;  
padding.Right = 70;  
var top = padding.Top;  
padding.Top = 50;
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

[HMI Unified Software object \(Page 272\)](#)

[Working with basic screen items \(Page 317\)](#)

7.18.9.3 Element screen items

Working with element screen items

Introduction

You can perform the following tasks with element screen items while using TIA Portal Openness:

- Creating screen item (element)
- Deleting screen item (element)
- Enumerating screen item (element)

The following screen items are supported by the TIA Portal Openness . To access the screen item related classes and datatype, use the following namespace and types:

Table 7-2 Types and Namespace (s)

Screen Item	Class	Namespace (s)
IO Field	HmiIOField	Siemens.Engineering.HmiUnified.ModernUI.Widgets
Button	HmiButton	Siemens.Engineering.HmiUnified.ModernUI.Base
Switch	HmiToggleSwitch	
Checkbox	HmiCheckBoxGroup	
Bar	HmiBar	
Gauge	HmiGauge	
Slider	HmiSlider	
RadioButton	HmiRadioButtonGroup	
List Box	HmiListBox	
Clock	HmiClock	
Text Box	HmiTextBox	

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)

Program code

You can modify and use the following program code example while working with element screen items.

Creating screen items (Element)

Modify the following program code to create element screen items:

```
HmiSoftware hmiSoftware = GetHMISoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screen[0].ScreenItems;
HmIIOField hmIIOField = screenitems.Create<HmIIOField>("ScreenItem_1");
```

Recoverable Exception will be raised if the Create method is called with Screen item Name with which Screen item already exists in TIA

Note

The above example to applicable to all screen items defined in type table above. You need to change the HmIIOField with type(s) defined in Types and Namespace (s) table.

Deleting screen items (Element)

Modify the following program code to delete element screen item:

```
public void Delete()
//Case 1
HmiSoftware hmiSoftware = GetHMISoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
HmIIOField hmIIOField = screenitems.Create<HmIIOField>("ScreenItems_1");
if (hmIIOField != null)
{
hmIIOField.Delete();
}
//Case 2
HmiSoftware hmiSoftware = GetHMISoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
HmIIOField hmIIOField = screenitems.Create<HmIIOField>("ScreenItem_1");
IEngineeringObject ObjhmIIOFieldEnggObj = hmIIOField;
if (ObjhmIIOFieldEnggObj != null)
{
ObjhmIIOFieldEnggObj.Invoke("Delete", null);
}
```

Note

The above example to applicable to all screen items defined in type table above. You need to change the HmIIOField with type(s) defined in Types and Namespace (s) table.

Enumerating screen items (Element)

Modify the following program code to enumerate all Screen Items of Screen.

```
HmiSoftware hmiSoftware = GetHMISoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
foreach (var item in screenitems)
{
//work with screenitems
}
```

Modify the following program code to searching Screen item from Screen items list on the basis of name:

```
HmiScreenItemBase screenitems = hmiSoftware.Screens[0].ScreenItems.Find("ScreenItems_1");
```

Modify the following program code to get Screen item from Screenitems list on basis of index:

```
HmiScreenItemBase screenitem = hmiSoftware.Screens[0].ScreenItems[0];
```

Modify the following program code to checking a particular screen item exist in Screen item list by using Contains method:

```
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
HmiIOField hmiIOField = screenitems.Create<HmiIOField>("ScreenItems_1");
bool isexists = screenitems.Contains(hmiIOField);
```

Note

The above example to applicable to all screen items defined in type table above. You need to change the HmiIOField with type(s) defined in Types and Namespace (s) table.

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

[HMI Unified Software object \(Page 272\)](#)

Accessing properties of IO Field

Introduction

You can use the TIA Portal Openness to get and set properties of IO field.

The following IO Field properties are supported in element screen items:

Property Name	Property Type	Accessibility
AlternateBackColor	Color	False
Enabled	Bool	False
AlternateBorderColor	Color	False
Authorization	String	True
BackColor	Color	False
BorderColor	Color	False
BorderWidth	byte	False
CurrentQuality	HmiQuality	True
Font	HmiFontPart	False
ForegroundColor	Color	False
Height	uint	False
HorizontalTextAlignment	HmiHorizontalAlignment	False
IOFieldType	HmiIOFieldType	False
Top	int	False
Left	int	False
Rotation	short	False
VisualizeQuality	bool	False
TextTrimming	HmiTextTrimming	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
Name	string	False
OutputFormat	string	False
Padding	HmiPaddingPart	True
RotationCenterPlacement	HmiRotationCenterPlacement	False
ProcessValue	string	False
TabIndex	ushort	False
ToolTipText	MultilingualText	False
VerticalTextAlignment	HmiVerticalAlignment	False
Visible	bool	False
Width	uint	False
Require Explicit Release	bool	False
InputBehavior	HmiInputBehaviorPart	True

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)

- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with basic screen items (Page 340)

Program code

Modify the following code to access basic properties of IO Field:

```
//Name
var name = iofield.Name;
iofield.Name = "DefaultName";
//AlternateBackColor
var altbackcolor = iofield.AlternateBackColor;
iofield.AlternateBackColor = Color.Beige;
//Height
var height = iofield.Height;
iofield.Height = 100;
```

Modify the following program code to access multilingual property:

```
//ToolTipText
HmiIOField ioField = createdscreen.ScreenItems.Create<HmiIOField>("hmiIOField");
string culture = "en-US";
Language lang = Tiaproject.LanguageSettings.Languages.Find(new CultureInfo(culture));
MultilingualText mltp = ioField.ToolTipText;
MultilingualTextItemComposition textItemComp = mltp.Items;
MultilingualTextItem mlttextitem = textItemComp.Find(lang);
mlttextitem.Text = "CommentInEnglish";
```

Modify the following program code to access other typical properties:

```
//Padding
var padding = iofield.Padding;
var bottom = padding.Bottom;
padding.Bottom = 50;
padding.Left = 60;
padding.Right = 70;
padding.Top = 50;
```

See also

- Connecting to the TIA Portal (Page 76)
- Opening a project (Page 106)
- HMI Unified Software object (Page 272)
- Working with element screen items (Page 340)

Accessing properties of button

Introduction

You can use the TIA Portal Openness to get and set properties of button in element screen items.

The following button properties are supported in element screen items:

Property Name	Property Type	Accessibility
AlternateBackgroundColor	Color	False
AlternateBorderColor	Color	False
Authorization	String	True
BackgroundColor	Color	False
BorderColor	Color	False
BorderWidth	byte	False
Contents	HmiContentPart	False
CurrentQuality	HmiQuality	True
Enabled	bool	False
Font	HmiFontPart	False
ForegroundColor	Color	False
Graphic	string	False
Height	uint	False
Top	int	False
Left	int	False
Rotation	short	False
VisualizeQuality	bool	False
RotationCenterX	float	False
RotationCenterY	float	False
Style Item Appearance	HmiStyleItem Appearance	False
Opacity	float	False
Name	string	False
Padding	HmiPaddingPart	False
RotationCenterPlacement	HmiRotationCenterPlacement	False
TabIndex	ushort	False
Text	MultiLingualText	True
ToolTipText	MultilingualText	True
Visible	bool	False
Width	uint	False
Require Explcit	bool	False
AlternateGraphic	string	False
AlternateText	MultiLingualText	False

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with element screen items (Page 340)

Program code

Modify the following program code to access basic properties of button:

```
//Name
var name = hmiButton.Name;
hmiButton.Name = "Default Value";
//AlternateBackColor
var altbackcolor = hmiButton.AlternateBackColor;
hmiButton.AlternateBackColor = Color.Beige;
//Height
var height = hmiButton.Height;
hmiButton.Height = 100;
```

Modify the following program code to access multilingual property:

```
//ToolTipText
string culture = "en-US";
Language lang = Tiaproject.LanguageSettings.Languages.Find(new CultureInfo(culture));
MultilingualText mltprop = hmiButton.ToolTipText;
MultilingualTextItemComposition textItemComp = mltprop.Items;
MultilingualTextItem mlttextitem = hmiButton.Find(lang);
mlttextitem.Text = "TestforMultilingualProperty";
```

Modify the following program code to access other typical property:

```
//Padding
var padding = hmiButton.Padding;
var bottom = padding.Bottom;
padding.Bottom = 50;
padding.Left = 60;
padding.Right = 70;
padding.Top = 50;
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

[HMI Unified Software object \(Page 272\)](#)

[Working with element screen items \(Page 340\)](#)

Accessing properties of switch

Introduction

You can use the TIA Portal Openness to get or set properties of switch in element screen items.

The following switch properties are supported in element screen items:

Property Name	Property Type	Accessibility
AlternateBackgroundColor	Color	False
AlternateBorderColor	Color	False
IsAlternateState	bool	False
Authorization	string	True
BackgroundColor	Color	False
BorderColor	Color	False
BorderWidth	byte	False
Contents	HmiContentPart	False
CurrentQuality	HmiQuality	True
Enabled	bool	False
Font	HmiFontPart	False
ForegroundColor	Color	False
Height	uint	False
Top	int	False
Left	int	False
Rotation	short	False
Text	MultiLingualText	True
VisualizeQuality	bool	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
Name	string	False
Padding	HmiPaddingPart	False
RotationCenterPlacement	HmiRotationCenterPlacement	False
TabIndex	ushort	False
ToolTipText	MultilingualText	False
Visibility	bool	False
Width	uint	False
Require Explixit	bool	False
Graphic	string	False
AlternateGraphic	string	False
AlternateText	MultiLingualText	True

Property Name	Property Type	Accessibility
Padding	HmiPaddingPart	True
SystemItemClass	HmiButtonStyleItemClass	True

Note

As StyleItemClass property is ReadOnly, thus Get Operation is supported for it and Set Operation will throw an exception.

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with element screen items (Page 340)

Program code

Modify the following program code to access basic properties of switch:

```
//Name
var name = toggleswitch.Name;
toggleswitch.Name = "Default Value";
//AlternateBackColor
var altbackcolor = toggleswitch.AlternateBackColor;
toggleswitch.AlternateBackColor = Color.Beige;
//Height
var height = toggleswitch.Height;
toggleswitch.Height = 100;
```

Modify the following program code to access multilingual property:

```
//ToolTipText
var tooltip = toggleswitch.ToolTipText;
var tooltiptext = toggleswitch.ToolTipText.Items[0].Text;
toggleswitch.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Modify the following program code to access other typical property of switch:

```
//Font
var font = toggleswitch.Font;
var italic = font.Italic;
font.Italic = false;
var fontname = font.FontName;
font.FontName = HmiFontName.SimSun;
var size = font.Size;
font.Size = 10.2f;
var stike = font.StrikeOut;
font.StrikeOut = true;
var underline = font.Underline;
font.Underline = false;
font.Bold = true;
```

See also

- [Connecting to the TIA Portal \(Page 76\)](#)
- [Opening a project \(Page 106\)](#)
- [HMI Unified Software object \(Page 272\)](#)
- [Working with element screen items \(Page 340\)](#)

Accessing properties of check box

Introduction

You can use the TIA Portal Openness to get or set properties of checkbox in element screen items.

The following check box properties are supported in element screen items:

Property Name	Property Type	Accessibility
AlternateBackgroundColor	Color	False
AlternateBorderColor	Color	False
Authorization	string	False
BackgroundColor	Color	False
BorderColor	Color	True
BorderWidth	byte	False
Content	HmiContentPart	False
CurrentQuality	HmiQuality	False
Enabled	bool	False
Font	HmiFontPart	True
ForegroundColor	Color	False
Height	uint	False
Top	int	False

7.18 Functions for accessing the data of an HMI device (Unified)

Property Name	Property Type	Accessibility
Left	int	False
Name	string	False
Rotation	short	False
VisualizeQuality	bool	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
Process value	string	False
RotationCenterPlacement	HmiRotationCenterPlacement	False
SelectionItemHeight	uint16	False
SelectionItems	HmiSelectionItemPartComposition	True
TabIndex	ushort	False
ToolTipText	MultilingualText	True
Visible	bool	False
Width	uint	False
Require Explixit Release	bool	False
Padding	HmiPaddingPart	False
SelectionPosition	HmiHorizontalAlignment	False

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with element screen items (Page 340)

Program code

Modify the following program code to access basic properties of Checkbox:

```
//Name
var name = chkbox.Name;
chkbox.Name = "DefaultName";
//AlternateBackColor
var altpbackcolor = chkbox.AlternateBackColor;
chkbox.AlternateBackColor = Color.Beige;
//Height
var height = chkbox.Height;
chkbox.Height = 100;
```

Modify the following program code to access multilingual property:

```
//ToolTipText
var tooltip = chkbox.ToolTipText;
var tooltiptext = chkbox.ToolTipText.Items[0].Text;
chkbox.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Modify the following program code to access other typical property of checkbox:

```
//Font
var font = chkbox.Font;
var italic = font.Italic;
font.Italic = false;
var fontname = font.FontName;
font.FontName = HmiFontName.SimSun;
var size = font.Size;
font.Size = 10.2f;
var stike = font.StrikeOut;
font.StrikeOut = true;
var underline = font.Underline;
font.Underline = false;
font.Bold = true;
//Selection Items
var selectionItem = chkbox.SelectionItems;
var newselectionitem = selectionItem.Create("NewSelectionItem");
var graphic = newselection.Graphic;
newselectionitem.Graphic = "abcd";
newselectionitem.Isselected = false;
newselectionitem.Text.Item[0].Text = "Testformultilingual";
```

See also

- [Connecting to the TIA Portal \(Page 76\)](#)
- [Opening a project \(Page 106\)](#)
- [HMI Unified Software object \(Page 272\)](#)
- [Working with element screen items \(Page 340\)](#)

Accessing properties of bar

Introduction

You can use the TIA Openness API V16 to get and set bar properties in element screen items.

The following bar properties are supported in element screen items:

Property Name	Property Type	Accessibility
AlternateBackgroundColor	Color	False
AlternateBorderColor	Color	False

7.18 Functions for accessing the data of an HMI device (Unified)

Property Name	Property Type	Accessibility
Authorization	string	True
BackgroundColor	Color	False
BorderColor	Color	False
BarMode	HmiBarMode	False
BorderWidth	byte	False
CurrentQuality	HmiQuality	True
Enabled	bool	False
Font	HmiFontPart	False
Label	HmiTextPart	
Height	uint	False
StraightScale	HmiStraightScalePart	
Top	int	False
Left	int	False
Name	string	False
NormalRangeColor	Color	False
OriginValue	double	False
OutputFormat	string	False
PeakIndicators	HmiPeakIndicator	False
Process Value	string	False
ProcessValueIndicatorBackColor	Color	False
ProcessValueIndicatorForeColor	Color	False
ProcessValueIndicatorMode	HmiProcessIndicatorMode	False
RotationAngle	short	False
VisualizeQuality	bool	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
RotationCenterPlacement	HmiRotationCenterPlacement	False
ScaleBackColor	uint16	False
ScaleForeColor	Color	False
ShowTrendIndicator	bool	False
Visible	bool	False
Width	uint	False
TabIndex	ushort	False
TrendIndicateColor	Color	False
ToolTipText	MultilingualText	False
Require Explicit Release	bool	False
Title	HmiTextPart	True

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with element screen items (Page 340)

Program code

Modify the following program code to access basic properties of bar:

```
//Name
var name = bar.Name;
bar.Name = "Default Value";
//RotationCenterPlacement
var rotation = bar.RotationCenterPlacement;
bar.RotationCenterPlacement = HmiRotationCenterPlacement.AbsoluteToContainer;
//Width
var width = bar.Width;
bar.Width = 100;
```

Modify the following program code to access multilingual property:

```
//ToolTipText
var tooltip = bar.ToolTipText;
var tooltiptext = bar.ToolTipText.Items[0].Text;
bar.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Modify the following program code to access other typical property of bar:

```
//Font
var title = bar.Title;
title.Text.items[0].Text = "teststing";
title.Visible = false;
title.Forecolor = Color.Black;
var font = title.Font;
var italic = font.Italic;
font.Italic = false;
var fontname = font.FontName;
font.FontName = HmiFontName.SimSun;
var size = font.Size;
font.Size = 10.2f;
var stike = font.StrikeOut;
font.StrikeOut = true;
var underline = font.Underline;
font.Underline = false;
font.Bold = true;
```

See also

- [Connecting to the TIA Portal \(Page 76\)](#)
- [Opening a project \(Page 106\)](#)
- [HMI Unified Software object \(Page 272\)](#)
- [Working with element screen items \(Page 340\)](#)

Accessing properties of gauge

Introduction

You can use the TIA Portal Openness to get or set properties of gauge in element screen items.

The following gauge properties are supported in element screen items:

Property Name	Property Type	Accessibility
AlternativebackgroundBackColor	Color	False
AlternativeBorderColor	Color	False
Authorization	string	True
BackgroundColor	Color	False
BorderColor	Color	False
BorderWidth	byte	False
RelativeToOrigin	bool	False
CurrentQuality	HmiQuality	True
Enabled	bool	False
Title	HmiTextPart	True
Font	HmiFontPart	False

Property Name	Property Type	Accessibility
Label	HmiTextPart	True
Height	uint	False
Top	int	False
Left	int	False
Name	string	False
NormalRangeColor	Color	False
OriginValue	double	False
OutputFormat	string	False
PeakIndicators	HmiPeakIndicator	False
Process Value	string	False
ProcessValueIndicatorBackColor	Color	False
ProcessValueIndicatorForeColor	Color	False
ProcessValueIndicatorMode	HmiProcessIndicatorMode	False
Rotation	short	False
VisualizeQuality	bool	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
RotationCenterPlacement	HmiRotationCenterPlacement	False
ScaleBackgroundColor	Color	False
ScaleForegroundColor	Color	False
TrendIndicatorColor	Color	False
Visible	bool	False
Width	uint	False
TabIndex	ushort	False
ShowTrendIndicator	bool	False
ToolTipText	string	False
RequireExplcit	bool	False
CurvedScale	HmiCurvedScalePart	True

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with element screen items (Page 340)

Program code

Modify the following program code to access basic properties of gauge:

```
//Name  
var name = gauge.Name;  
gauge.Name = "Default Value";  
//RotationalCenterPlacement  
var rotation = gauge.RotationCenterPlacement;  
gauge.RotationCenterPlacement = HmiRotationCenterPlacement.AbsoluteToContainer;  
//Width  
var width = gauge.Width;  
gauge.Width = 100;
```

Modify the following program code to access multilingual property:

```
//ToolTipText  
var tooltip = gauge.ToolTipText;  
var tooltiptext = gauge.ToolTipText.Items[0].Text;  
gauge.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Modify the following program code to access other typical property of gauge:

```
//Font  
var title = gauge.Title;  
title.Text.items[0].Text = "teststing";  
title.Visible = false;  
title.Forecolor = Color.Black;  
var font = title.Font;  
var italic = font.Italic;  
font.Italic = false;  
var fontname = font.FontName;  
font.FontName = HmiFontName.SimSun;  
var size = font.Size;  
font.Size = 10.2f;  
var stike = font.StrikeOut;  
font.StrikeOut = true;  
var underline = font.Underline;  
font.Underline = false;  
font.Bold = true;
```

See also

- [Connecting to the TIA Portal \(Page 76\)](#)
- [Opening a project \(Page 106\)](#)
- [HMI Unified Software object \(Page 272\)](#)
- [Working with element screen items \(Page 340\)](#)

Accessing properties of slider

Introduction

You can use the TIA Portal Openness to get and set properties of slider in element screen items.

The following slider properties are supported in element screen items:

Property Name	Property Type	Accessibility
AlternativeBackColor	Color	False
AlternativeBorderColor	Color	False
Authorization	string	True
BackgroundColor	Color	False
BorderColor	Color	False
BorderWidth	byte	False
BarMode	HmiBarMode	False
RelativeToOrigin	bool	False
CurrentQuality	HmiQuality	True
Enabled	bool	False
Title	HmiTextPart	True
Font	HmiFontPart	False
Label	HmiTextPart	True
Height	uint	False
Top	int	False
Left	int	False
Name	string	False
NormalRangeColor	Color	False
OriginValue	double	False
OutputFormat	string	False
PeakIndicators	HmiPeakIndicator	False
Process Value	string	False
ProcessValueIndicatorBackColor	Color	False
ProcessValueIndicatorForeColor	Color	False
ProcessValueIndicatorMode	HmiProcessIndicatorMode	False
Rotation	short	False
VisualizeQuality	bool	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
RotationCenterPlacement	HmiRotationCenterPlacement	False
ScaleBackgroundColor	Color	False
ScaleForegroundColor	Color	False
TrendIndicatorColor	bool	False
Visible	bool	False
Width	uint	False

7.18 Functions for accessing the data of an HMI device (Unified)

Property Name	Property Type	Accessibility
ValuePosition	HmiSimplePosition	False
WriteDuringChange(Write process value Immediatly)	bool	False
TabIndex	ushort	False
ShowTrendIndicator	bool	False
ToolTipText	string	False
ShowValue	bool	False
Require Explixit Release	bool	False
ThumbBackColor	color	False
ThumbForeColor	color	False
StraightScale	HmiStraightScalePart	True

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with element screen items (Page 340)

Program code

Modify the following program code to access basic properties of slider:

```
//Name
var name = slider.Name;
slider.Name = "Default Value";
//RotationalCenterPlacement
var rotation = slider.RotationCenterPlacement;
slider.RotationCenterPlacement = HmiRotationCenterPlacement.AbsoluteToContainer;
//Width
var width = slider.Width;
slider.Width = 100;
```

Modify the following program code to access multilingual property:

```
//ToolTipText
var tooltip = slider.ToolTipText;
var tooltiptext = slider.ToolTipText.Items[0].Text;
slider.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Modify the following program code to access other typical properties of slider:

```
//Font
var title = slider.Title;
title.Text.items[0].Text = "teststing";
title.Visible = false;
title.Forecolor = Color.Black;
var font = title.Font;
var italic = font.Italic;
font.Italic = false;
var fontname = font.FontName;
font.FontName = HmiFontName.SimSun;
var size = font.Size;
font.Size = 10.2f;
var stike = font.StrikeOut;
font.StrikeOut = true;
var underline = font.Underline;
font.Underline = false;
font.Bold = true;
```

See also

- [Connecting to the TIA Portal \(Page 76\)](#)
- [Opening a project \(Page 106\)](#)
- [HMI Unified Software object \(Page 272\)](#)
- [Working with element screen items \(Page 340\)](#)

Accessing properties of radio button

Introduction

You can use the TIA Portal Openness to get and set properties of radio button in element screen items.

The following radio button properties are supported in element screen items:

Property Name	Property Type	Accessibility
AlternateBackgroundColor	Color	False
AlternateBorderColor	Color	False
Authorization	string	True
BackgroundColor	Color	False
BorderColor	Color	False
BorderWidth	byte	False
Content	HmiContentPart	False
CurrentQuality	HmiQuality	True
Enabled	bool	False
Font	HmiFontPart	False

7.18 Functions for accessing the data of an HMI device (Unified)

Property Name	Property Type	Accessibility
ForegroundColor	Color	False
Height	uint	False
Top	int	False
Left	int	False
Name	string	False
Process Value	string	False
Rotation	short	False
VisualizeQuality	bool	False
RotationCenterX	float	False
RotationCenterY	float	False
SelectionItemHeight	uint16	False
SelectionItems	IList<IHmiSelectionItemPart>	True
SelectorPosition	HmiHorizontalAlignment	
Opacity	float	False
RotationCenterPlacement	HmiRotationCenterPlacement	False
Visible	bool	False
Width	uint	False
TabIndex	ushort	False
ToolTipText	string	False
Require Explixit Release	bool	False
Padding	HmiPaddingPart	True

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with element screen items (Page 340)

Program code

Modify the following program code to access basic properties of radio button:

```
//Name
var name = radio.Name;
radio.Name = "Default Value";
//AlternateBackColor
var altbackcolor = radio.AlternateBackColor;
radio.AlternateBackColor = Color.Beige;
//Height
var height = radio.Height;
radio.Height = 100;
```

Modify the following program code to access multilingual property:

```
//ToolTipText
var tooltip = radio.ToolTipText;
var tooltiptext = radio.ToolTipText.Items[0].Text;
radio.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Modify the following program code to access other typical properties of radio button:

```
//Font
var font = radio.Font;
var italic = font.Italic;
font.Italic = false;
var fontname = font.FontName;
font.FontName = HmiFontName.SimSun;
var size = font.Size;
font.Size = 10.2f;
var stike = font.StrikeOut;
font.StrikeOut = true;
var underline = font.Underline;
font.Underline = false;
font.Bold = true;
//Selection Items
var selectionItem = radio.SelectionItems;
var newselectionitem = selectionItem.Create("NewSelectionItem");
var graphic = newselectionitem.Graphic;
newselectionitem.Graphic = "abcd";
newselectionitem.IsSelected = false;
newselectionitem.Text.Items[0].Text = "Testformultilingual";
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

[HMI Unified Software object \(Page 272\)](#)

[Working with element screen items \(Page 340\)](#)

Accessing properties of listbox

Introduction

You can use the TIA Portal Openness to get and set properties of listbox in element screen items.

The following listbox properties are supported in element screen items:

Property Name	Property Type	Accessibility
AlternateBackgroundColor	Color	False
AlternateBorderColor	Color	False
Authorization	string	True
BackgroundColor	Color	False
BorderColor	Color	False
BorderWidth	byte	False
Content	HmiContentPart	False
CurrentQuality	HmiQuality	True
Enabled	bool	False
Font	HmiFontPart	False
ForeGroundColor	Color	False
Top	int	False
Left	int	False
Name	string	False
Process Value	string	False
Rotation	short	False
VisualizeQuality	bool	False
RotationCenterX	float	False
RotationCenterY	float	False
SelectionItemHeight	uint16	False
SelectionItems	IList<IHmiSelectionItemPart>	True
SelectionMode	HmiSelectionMode	False
SelectorPosition	HmiHorizontalAlignment	
Opacity	float	False
RotationCenterPlacement	HmiRotationCenterPlacement	False
Visible	bool	False
Width	uint	False
TabIndex	ushort	False
ToolTipText	string	False
Require Explixit Release	bool	False
Padding	HmiPaddingPart	True

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with element screen items (Page 340)

Program code

Modify the following program code to access basic properties of list box:

```
//Name
var name = listbox.Name;
listbox.Name = "Default Value";
//AlternateBackColor
var altbackcolor = listbox.AlternateBackColor;
listbox.AlternateBackColor = Color.Beige;
//Height
var height = listbox.Height;
listbox.Height = 100;
```

Modify the following program code to access multilingual property:

```
//ToolTipText
var tooltip = listbox.ToolTipText;
var tooltiptext = listbox.ToolTipText.Items[0].Text;
listbox.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Modify the following program code to access other typical property of list box:

```
//Font
var font = listbox.Font;
var italic = font.Italic;
font.Italic = false;
var fontname = font.FontName;
font.FontName = HmiFontName.SimSun;
var size = font.Size;
font.Size = 10.2f;
var strike = font.StrikeOut;
font.StrikeOut = true;
var underline = font.Underline;
font.Underline = false;
font.Bold = true;
//Selection Items
var selectionItem = listbox.SelectionItems;
var newselectionitem = selectionItem.Create("NewSelectionItem");
var graphic = newselectionitem.Graphic;
newselectionitem.Graphic = "abcd";
newselectionitem.Isselected = false;
newselectionitem.Text.Item[0].Text = "Testformultilingual";
```

See also

- [Connecting to the TIA Portal \(Page 76\)](#)
- [Opening a project \(Page 106\)](#)
- [HMI Unified Software object \(Page 272\)](#)
- [Working with element screen items \(Page 340\)](#)

Accessing properties of textbox

Introduction

You can use the TIA Portal Openness to get and set properties of textbox in element screen items.

The following textbox properties are supported in element screen items:

Property Name	Property Type	Accessibility
ReadOnly	bool	False
TextWrapping	HmiTextWrapping	False
VerticalTextAlignment	HmiVerticalAlignment	False
HorizontalTextAlignment	HmiHorizontalAlignment	False
TextTrimming	HmiTextTrimming	False
ForeColor	Color	False
BorderColor	Color	False
AlternateBorderColor	Color	False

Property Name	Property Type	Accessibility
Authorization	string	False
BackColor	Color	False
BorderWidth	byte	False
CurrentQuality	HmiQuality	True
Enabled	bool	False
Font	HmiFontPart	False
Height	uint	False
Top	int	False
Left	int	False
Name	string	False
RotationAngle	short	False
VisualizeQuality	bool	False
AlternateBackColor	Color	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
RotationCenterPlacement	HmiRotationCenterPlacement	False
Visible	bool	False
Width	uint	False
TabIndex	ushort	False
ToolTipText	string	False
Require Explicit Release	bool	False
Text	MultilingualText	True
Padding	HmiPaddingPart	True

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with element screen items (Page 340)

Program code

Modify the following program code to access basic property of textbox:

```
//Name  
var name = textbox.Name;  
textbox.Name = "Default Value";  
//BorderColor  
var bordercolor = textbox.BorderColor;  
textbox.BorderColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);  
//BorderWidth  
var borderwidth = textbox.BorderWidth;  
textbox.BorderWidth = 10;
```

Modify the following program code to access multilingual property:

```
//ToolTipText  
var tooltip = textbox.ToolTipText;  
var tooltiptext = textbox.ToolTipText.Items[0].Text;  
textbox.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Modify the following program code to access other typical property of textbox:

```
//Font  
var font = textbox.Font;  
var italic = font.Italic;  
font.Italic = false;  
var fontname = font.FontName;  
font.FontName = HmiFontName.SimSun;  
var size = font.Size;  
font.Size = 10.2f;  
var stike = font.StrikeOut;  
font.StrikeOut = true;  
var underline = font.Underline;  
font.Underline = false;  
font.Bold = true;
```

See also

- Connecting to the TIA Portal (Page 76)
- Opening a project (Page 106)
- HMI Unified Software object (Page 272)
- Working with element screen items (Page 340)

Accessing properties of clock

Introduction

You can use the TIA Portal Openness to get and set the properties of clock in element screen item.

The following clock properties are supported in element screen item:

Property Name	Property Type	Accessibility
AlternativeBackgroundColor	Color	False
AlternativeBorderColor	Color	False
BorderColor	Color	False
Authorization	string	True
BackgroundColor	Color	False
BorderWidth	byte	False
CurrentQuality	HmiQuality	True
DialBackColor	Color	False
DialLabelColor	Color	False
DialMode	HmiScaleMode	False
DialTickColor	Color	False
Enabled	bool	False
Height	uint	False
Top	int	False
Left	int	False
Name	string	False
VisualizeQuality	bool	False
RotationCenterX	float	False
RotationCenterY	float	False
ShowHours	bool	False
ShowMinutes	bool	False
ShowSeconds	bool	False
Opacity	float	False
TimeSource	string	False
Rotation	short	False
RotationCenterPlacement	HmiRotationCenterPlacement	False
Visibility	bool	False
Width	uint	False
TabIndex	ushort	False
Title	HmiTextPart	True
ToolTipText	MultiLingualText	False
Require Explixit	bool	False
DialLabelFont	HmiFontPart	True

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with element screen items (Page 340)

Program code

Modify the following program code to access basic property of Clock:

```
//Name
var name = clock.Name;
clock.Name = "Default Value";
//AlternateBackColor
var altbackcolor = clock.AlternateBackColor;
clock.AlternateBackColor = Color.Beige;
//Height
var height = clock.Height;
clock.Height = 100;
```

Modify the following program code to access multilingual property:

```
//ToolTipText
var tooltip = clock.ToolTipText;
var tooltiptext = clock.ToolTipText.Items[0].Text;
clock.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Modify the following program code to access other typical properties:

```
//DialLabelFont
var dialfont = clock.DialLabelFont;
dialfont.Italic = false;
var fontname = dialfont.FontName;
dialfont.FontName = HmiFontName.SimSun;
var size = dialfont.Size;
dialfont.Size = 10.2f;
var stike = dialfont.StrikeOut;
dialfont.StrikeOut = true;
var underline = dialfont.Underline;
dialfont.Underline = false;
dialfont.Bold = true;
//ShowHours
var showhour = clock.ShowHours;
clock.ShowHours = false;
```

Note

If during set of properties the set operation is not able to set the value then a Recoverable exception would be raised. Values beyond the ranges if not set will send a Recoverable exception or if set will put the property in invalid state.

See also

- [Connecting to the TIA Portal \(Page 76\)](#)
- [Opening a project \(Page 106\)](#)
- [HMI Unified Software object \(Page 272\)](#)
- [Working with element screen items \(Page 340\)](#)

7.18.9.4 Control screen items**Working with control screen items****Introduction**

You can perform the following tasks with control screen items while using TIA Portal Openness:

- Creating screen items (controls)
- Deleting screen items (controls)
- Enumerating screen items (controls)

The following screen items are supported by the TIA Portal Openness. To access the screen item related classes and datatype, use the following namespace and types:

Table 7-3 Types and Namespace (s)

Screen Item	Class	Namespace (s)
Alarm Control	HmiAlarmControl	Siemens.Engineering.HmiUnified.ModernUI.Controls
Media Player	HmiMediaControl	Siemens.Engineering.HmiUnified.ModernUI.Base
Screen Window	HmiScreenWindow	
Trend Control	HmiTrendControl	
Trend Companion	HmiTrendCompanion	
Process Control	HmiProcessControl	
Function Trend Control	HmiFunctionTrendControl	
Web Control	HmiWebControl	
Parameter Set Control	HmiDetailedParameter-Control	

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)

Program code

You can modify and use the following program code example while working with control screen items.

Creating screen items (Controls)

Modify the following program code to create control screen items:

```
HmiSoftware hmiSoftware = GetHMISoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
HmiAlarmControl hmiAlarmControl = screenitems.Create<HmiAlarmControl>("ScreenItem_1");
```

Recoverable Exception will be raised if the Create method is called with Screen item Name with which Screen item already exists in TIA

Note

The above example is applicable to all screen items defined in type table above. You need to change the HmiAlarmControl with type(s) defined in Types and Namespace(s) table.

Deleting screen items (Controls)

Modify the following program code to delete control screen items:

```
public void Delete()
//Case 1
HmiSoftware hmiSoftware = GetHMISoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
HmiAlarmControl hmiAlarmControl = screenitems.Create<HmiAlarmControl>("ScreenItem_1");
if (hmiAlarmControl != null)
{
hmiAlarmControl Delete();
}
//Case 2
HmiSoftware hmiSoftware = GetHMISoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
HmiAlarmControl hmiAlarmControl = screenitems.Create<HmiAlarmControl>("ScreenItem_1");
IEngineeringObject ObjhmiAlarmControlEnggObj = hmiAlarmControl;
if (ObjhmiAlarmControlEnggObj != null)
{
ObjhmiAlarmControlEnggObj.Invoke("Delete", null);
}
```

Note

The above example is applicable to all screen items defined in type table above. You need to change the HmiAlarmControl with type(s) defined in Types and Namespace(s) table.

Enumerating screen items (Control)

Modify the following program code to enumerate all Screen Items of Screen:

```
HmiSoftware hmiSoftware = GetHMISoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
foreach (var item in screenitems)
{
//work with screenitems
}
```

Modify the following program code to search Screen item from Screen items list on the basis of name:

```
HmiScreenItemBase screenitems = hmiSoftware.Screens[0].ScreenItems.Find("ScreenItems_1");
```

Modify the following program code to get Screen item from Screenitems list on basis of index:

```
HmiScreenItemBase screenitem = hmiSoftware.Screens[0].ScreenItems[0];
```

7.18 Functions for accessing the data of an HMI device (Unified)

Modify the following program code to checking a particular screen item exist in Screen item list by using Contains method:

```
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
HmiAlarmControl hmiAlarmControl = screenitems.Create<HmiAlarmControl>("ScreenItems_1");
bool isexists = screenitems.Contains(hmiAlarmControl);
```

Note

The above example is applicable to all screen items defined in type table above. You need to change the HmiAlarmControl with type(s) defined in Types and Namespace (s) table.

See also

- [Connecting to the TIA Portal \(Page 76\)](#)
- [Opening a project \(Page 106\)](#)
- [HMI Unified Software object \(Page 272\)](#)

Accessing properties of screen window

Introduction

You can use the TIA Portal Openness to get and set properties of screen window in control screen items.

The following screen window properties are supported in control screen items:

Property Name	Property Type	Accessibility
TabIntoWindow	bool	False
Screen	string	False
ScreenName	string	True
ScreenNumber	byte	True
System	byte	False
CurrentZoomFactor	double	False
VerticalScrollBarVisibility	HmiScrollBarVisibility	False
VerticalScrollBarPosition	int	False
HorizontalScrollBarVisibility	HmiScrollBarVisibility	False
HorizontalScrollBarPosition	int	False
Adaption	HmiScreenWidthAdaption	False
InteractiveZooming	bool	True
Caption	MultilingualProperty	False
WindowFlags	HmiWindowFlag	False
Icon	string	False
Top	int	False

Property Name	Property Type	Accessibility
Left	int	False
Width	uint	False
Height	uint	False
CurrentQuality	HmiQuality	True
RequireExplicitRelease	bool	False
Authorization	string	False
Name	string	False
Visible	bool	False
Enabled	bool	False
TabIndex	ushort	False

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with control screen items (Page 369)

Program code

Modify the following program code to access basic properties of screen window:

```
//Width
var autoplay = hmiscreenwindow.Width;
hmiscreenwindow.Width = 1;
//HorizontalScrollBarVisibility
var horizontalscrollbarvisibilityValue = hmiscreenwindow.HorizontalScrollBarVisibility;
hmiscreenwindow.HorizontalScrollBarVisibility = HmiScrollBarVisibility.Visible;
//Name
var name = hmiscreenwindow.Name;
hmiscreenwindow.Name = "DefaultName";
```

Modify the following program code to access multilingual property:

```
//Caption
var caption = hmiscreenwindow.Caption.Items[0].Text;
hmiscreenwindow.Caption.Items[0].Text = "<body><p>TestforMultilingualProperty</p></body>";
```

See also

- Connecting to the TIA Portal (Page 76)
- Opening a project (Page 106)
- HMI Unified Software object (Page 272)
- Working with control screen items (Page 369)

Accessing properties of web control

Introduction

You can use the TIA Portal Openness to get and set properties of web control in control screen items.

The following web control properties are supported in control screen items.

Property Name	Property Type	Accessibility
Url	string	False
BackColor	Color	False
Caption	MultilingualProperty	False
WindowFlags	HmiWindowFlag	False
Icon	string	False
Top	Int	False
Left	int	False
Width	uint	False
Height	uint	False
CurrentQuality	HmiQuality	False
RequireExplicitRelease	bool	False
Authorization	string	False
Name	string	False
Visible	bool	False
Enabled	bool	False
TabIndex	ushort	False
ToolBar	HmiToolBarPart	True
StatusBar	HmiStatusBarPart	True

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)

- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with control screen items (Page 369)

Program code

Modify the following program code to access basic properties of web control:

```
//Width
var autoplay = hmiwebcontrol.width;
hmiwebcontrol.Width = 1;
//BackColor
var backcolor = hmiwebcontrol.BackColor;
hmiwebcontrol.BackColor = Color.Beige;
//Name
var name = hmiwebcontrol.Name;
hmiwebcontrol.Name = "DefaultName";
```

Modify the following program code to access multilingual property:

```
//Caption
var caption = hmiwebcontrol.Caption.Items[0].Text;
hmiwebcontrol.Caption.Items[0].Text= "<body><p>TestforMultilingualProperty</p></body>";
```

Modify the following program code to access other typical properties:

```
//Status bar
var statusBar = hmiwebcontrol.StatusBar;
var backColor = statusBar.BackColor;
statusBar.BackColor = Color.Aqua;
var enabled = statusBar.Enabled;
statusBar.Enabled = true;
var visible = statusBar.Visible;
statusBar.Visible = true;
```

See also

- Connecting to the TIA Portal (Page 76)
- Opening a project (Page 106)
- HMI Unified Software object (Page 272)
- Working with control screen items (Page 369)

Accessing properties of parameter set control

Introduction

You can use the TIA Portal Openness to get and set properties of parameter set control in control screen items.

The following parameter set control properties are supported in control screen items.

Property Name	Property Type	Accessibility
ParameterSetTypeFixed	string	False
BackColor	Color	False
Caption	MultilingualProperty	False
WindowFlags	HmiWindowFlag	False
Icon	string	False
Top	int	False
Left	int	False
Width	uint	False
Height	uint	False
CurrentQuality	HmiQuality	True
RequireExplicitRelease	bool	False
Authorization	string	False
Name	string	False
Visible	bool	False
Enabled	bool	False
TabIndex	ushort	False
ToolBar	HmiToolBarPart	True
StatusBar	HmiStatusBarPart	True
EditMode	HmiEditMode	False
TimeZone	int	False
ParameterView	HmiDataGridViewPart	True
ForeColor	Color	False
SelectionBackColor	Color	False
SelectionForeColor	Color	False
Font	HmiFontPart	True

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)

- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with control screen items (Page 369)

Program code

Modify the following program code to access basic properties of parameter set control:

```
//Width
var autoplay = hmidetailedparametercontrol.Width;
hmidetailedparametercontrol.Width = 1;
//BackColor
var backcolor = hmidetailedparametercontrol.BackColor;
hmidetailedparametercontrol.BackColor = Color.Beige;
//Name
var name = hmidetailedparametercontrol.Name;
hmidetailedparametercontrol.Name = "Default";
```

Modify the following program code to access multilingual property:

```
//Caption
var caption = hmidetailedparametercontrol.Caption.Items[0].Text;
hmidetailedparametercontrol.Caption.Items[0].Text=
"<body><p>TestforMultilingualProperty</p></body>";
```

Modify the following program code to access other typical property:

```
//Status bar
var statusBar = hmidetailedparametercontrol.StatusBar;
var backColor = statusBar.BackColor;
statusBar.BackColor = Color.Aqua;
var enabled = statusBar.Enabled;
statusBar.Enabled = true;
var visible = statusBar.Visible;
statusBar.Visible = true;
```

See also

- [Connecting to the TIA Portal \(Page 76\)](#)
- [Opening a project \(Page 106\)](#)
- [HMI Unified Software object \(Page 272\)](#)
- [Working with control screen items \(Page 369\)](#)

Accessing properties of alarm control

Introduction

You can use the TIA Portal Openness to get and set properties of alarm control in control screen items.

The following alarm control properties are supported in control screen items.

Property Name	Property Type	Accessibility
UseAlarmColors	bool	False
Filter	string	False
AlwaysShowRecent	bool	False
AlarmSourceType	HmiAlarmSourceType	False
Caption	MultilingualProperty	False
WindowFlags	HmiWindowFlag	False
Icon	string	False
Top	Int	False
Left	int	False
Width	uint	False
Height	uint	False
CurrentQuality	HmiQuality	True
RequireExplicitRelease	bool	False
Authorization	string	False
Name	string	False
Visible	bool	False
Enabled	bool	False
TabIndex	ushort	False
ToolBar	HmiToolBarPart	True
StatusBar	HmiStatusBarPart	True
EditMode	HmiEditMode	False
TimeZone	int	False
AlarmDefinitionViewSetup	HmiVisibleAlarms	False
ActiveAlarmsViewSetup	HmiVisibleAlarms	False
SuppressFlashing	bool	False
AcknowledgmentFlashingRate	HmiFlashingRate	False
ResetFlashingRate	HmiFlashingRate	False
DefaultSortDirection	HmiSortDirection	False
BackColor	Color	False
AlarmView	HmiDataGridViewPart	True

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with control screen items (Page 369)

Program code

Modify the following program code to access basic properties of alarm control:

```
//Width
var autoplay = hmialarmcontrol.Width;
hmialarmcontrol.Width = 1;
//BackColor
var backcolor = hmialarmcontrol.BackColor;
hmialarmcontrol.BackColor = Color.Beige;
//Name
var name = hmialarmcontrol.Name;
hmialarmcontrol.Name = "DefaultName";
```

Modify the following program code to access multilingual property:

```
//Caption
var caption = hmialarmcontrol.Caption.Items[0].Text;
hmialarmcontrol.Caption.Items[0].Text = "<body><p>TestforMultilingualProperty</p></body>";
```

Modify the following program code to access other typical properties:

```
//Status bar
var statusBar = hmialarmcontrol.StatusBar;
var backColor = statusBar.BackColor;
statusBar.BackColor = Color.Aqua;
var enabled = statusBar.Enabled;
statusBar.Enabled = true;
var visible = statusBar.Visible;
statusBar.Visible = true;
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.18 Functions for accessing the data of an HMI device (Unified)

[HMI Unified Software object \(Page 272\)](#)

[Working with control screen items \(Page 369\)](#)

Accessing properties of trend companion

Introduction

You can use the TIA Portal Openness to get and set properties of trend companion in control screen items.

The following trend companion properties are supported in control screen items.

Property Name	Property Type	Accessibility
TrendCompanionMode	HmiTrendCompanionMode	False
UseSourceControlTrendColors	bool	False
ShowAlways	bool	False
UseSourceControlBackColor	bool	False
Caption	MultilingualProperty	False
WindowFlags	HmiWindowFlag	False
Icon	string	False
Top	int	False
Left	int	False
Width	uint	False
Height	uint	False
CurrentQuality	HmiQuality	True
RequireExplicitRelease	bool	False
Authorization	string	False
Name	string	False
Visible	bool	False
Enabled	bool	False
TabIndex	ushort	False
ToolBar	HmiToolBarPart	True
StatusBar	HmiStatusBarPart	True
SourceTrendControl	string	False
TimeZone	int	False
SnapToSourceControl	bool	False
TrendRulerView	HmiDataGridViewPart	True
TrendStatisticAreaView	HmiDataGridViewPart	True
BackColor	Color	False
TrendStatisticResultView	HmiDataGridViewPart	True

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with control screen items (Page 369)

Program code

Modify the following program code to access basic properties of trend companion:

```
//Width
var autoplay = hmitrendcompanion.Width;
hmitrendcompanion.Width = 1;
//BackColor
var backcolor = hmitrendcompanion.BackColor;
hmitrendcompanion.BackColor = Color.Beige;
//Name
var name = hmitrendcompanion.Name;
hmitrendcompanion.Name = "DefaultName";
```

Modify the following program code to access multilingual property access:

```
//Caption
var caption = hmitrendcompanion.Caption.Items[0].Text;
hmitrendcompanion.Caption.Items[0].Text = "<body><p>TestforMultilingualProperty</p></body>";
```

Modify the following program code to access other typical properties of trend companion:

```
//Status bar
var statusBar = hmitrendcompanion.StatusBar;
var backColor = statusBar.BackColor;
statusBar.BackColor = Color.Aqua;
var enabled = statusBar.Enabled;
statusBar.Enabled = true;
var visible = statusBar.Visible;
statusBar.Visible = true;
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

[HMI Unified Software object \(Page 272\)](#)

[Working with control screen items \(Page 369\)](#)

Accessing properties of trend control

Introduction

You can use the TIA Portal Openness to get and set properties of trend control in control screen items.

The following trend control properties are supported in control screen items.

Property Name	Property Type	Accessibility
ShowStatisticRulers	bool	False
AreaSpacing	ushort	False
Online	bool	False
ExtendRulerToAxis	bool	False
Caption	MultilingualProperty	False
WindowFlags	HmiWindowFlag	False
Icon	string	False
Top	int	False
Left	int	False
Width	uint	False
Height	uint	False
CurrentQuality	HmiQuality	True
RequireExplicitRelease	bool	False
Authorization	string	False
Name	string	False
Visible	bool	False
Enabled	bool	False
TabIndex	ushort	False
ToolBar	HmiToolBarPart	True
StatusBar	HmiStatusBarPart	True
Font	HmiFontPart	True
TimeZone	int	False
ShowRuler	bool	False
ShiftAxes	bool	False
BackColor	Color	False
Legend	HmiLegendPart	True
ForeColor	Color	False

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with control screen items (Page 369)

Program code

Modify the following program code to access basic properties of trend control:

```
//Width
var autoplay = hmitrendcontrol.Width;
hmitrendcontrol.Width = 1;
//BackColor
var backcolor = hmitrendcontrol.BackColor;
hmitrendcontrol.BackColor = Color.Beige;
//Name
var name = hmitrendcontrol.Name;
hmitrendcontrol.Name = "DefaultName";
```

Modify the following program code to access multilingual property access:

```
//Caption
var caption = hmitrendcontrol.Caption.Items[0].Text;
hmitrendcontrol.Caption.Items[0].Text = "<body><p>TestforMultilingualProperty</p></body>";
```

Modify the following program code to access other typical properties:

```
//Status bar
var statusBar = hmitrendcontrol.StatusBar;
var backColor = statusBar.BackColor;
statusBar.BackColor = Color.Aqua;
var enabled = statusBar.Enabled;
statusBar.Enabled = true;
var visible = statusBar.Visible;
statusBar.Visible = true;
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.18 Functions for accessing the data of an HMI device (Unified)

HMI Unified Software object (Page 272)

Working with control screen items (Page 369)

Accessing properties of media player

Introduction

You can use the TIA Portal Openness to get and set properties of media player in control screen items.

The following media player properties are supported in control screen items.

Property Name	Property Type	Accessibility
Url	string	False
AutoPlay	bool	False
VideoOutput	HmiVideoOutput	False
BackColor	Color	False
Caption	MultilingualProperty	False
WindowFlags	HmiWindowFlag	False
Icon	string	False
Top	int	False
Left	int	False
Width	uint	False
Height	uint	False
CurrentQuality	HmiQuality	True
RequireExplicitRelease	bool	False
Authorization	string	False
Name	string	False
Visible	bool	False
Enabled	bool	False
TabIndex	ushort	False
ToolBar	HmiToolBarPart	True
StatusBar	HmiStatusBarPart	True
EventHandlers	HmiMediaControlEventHandlerComposition	True

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)

- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with control screen items (Page 369)

Program code

Modify the following program code to access basic properties of media player:

```
//Autoplay
var autoplay = hmimediicontrol.AutoPlay;
hmimediicontrol.AutoPlay = true;
//BackColor
var backcolor = hmimediicontrol.BackColor;
hmimediicontrol.BackColor = Color.Beige;
//Name
var name = hmimediicontrol.Name;
hmimediicontrol.Name = "DefaultName";
```

Modify the following program code to access multilingual property:

```
//Caption
var caption = hmimediicontrol.Caption.Items[0].Text;
hmimediicontrol.Caption.Items[0].Text = "<body><p>TestforMultilingualProperty</p></body>";
```

Modify the following program code to access other typical property:

```
var statusBar = hmimediicontrol.StatusBar;
var backColor = statusBar.BackColor;
statusBar.BackColor = Color.Aqua;
var enabled = statusBar.Enabled;
statusBar.Enabled = true;
var visible = statusBar.Visible;
statusBar.Visible = true;
```

See also

- Connecting to the TIA Portal (Page 76)
- Opening a project (Page 106)
- HMI Unified Software object (Page 272)
- Working with control screen items (Page 369)

Accessing properties of process control

Introduction

You can use the TIA Portal Openness to get and set properties of process control in control screen items.

The following process control properties are supported in control screen items:

Property Name	Property Type	Accessibility
Online	bool	False
TimeStepSmoothingBase	HmiTimeRangeBase	False
TimeStepSmoothingFactor	int	False
BackColor	Color	False
Caption	MultilingualProperty	False
WindowFlags	HmiWindowFlag	False
Icon	string	False
Top	int	False
Left	int	False
Width	uint	False
Height	uint	False
CurrentQuality	HmiQuality	True
RequireExplicitRelease	bool	False
Authorization	string	False
Name	string	False
Visible	bool	False
Enabled	bool	False
TabIndex	ushort	False
ToolBar	HmiToolBarPart	True
StatusBar	HmiStatusBarPart	True
EditMode	HmiEditMode	False
TimeZone	int	False
ProcessView	HmiDataGridViewPart	True
EventHandlers	HmiProcessControlEventHandlerComposition	True

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)

- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with control screen items (Page 369)

Program code

Modify the following program code to access basic properties of process control:

```
//Width
var autoplay = hmiprocesscontrol.Width;
hmiprocesscontrol.Width = 1;
//BackColor
var backcolor = hmiprocesscontrol.BackColor;
hmiprocesscontrol.BackColor = Color.Beige;
//Name
var name = hmiprocesscontrol.Name;
hmiprocesscontrol.Name = "DefaultName";
```

Modify the following program code to access multilingual property:

```
//Caption
var caption = hmiprocesscontrol.Caption.Items[0].Text;
hmiprocesscontrol.Caption.Items[0].Text= "<body><p>TestforMultilingualProperty</p></body>";
```

Modify the following program code to access other typical properties:

```
//status bar
var statusBar = hmiprocesscontrol.StatusBar;
var backColor = statusBar.BackColor;
statusBar.BackColor = Color.Aqua;
var enabled = statusBar.Enabled;
statusBar.Enabled = true;
var visible = statusBar.Visible;
statusBar.Visible = true;
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

[HMI Unified Software object \(Page 272\)](#)

[Working with control screen items \(Page 369\)](#)

Accessing properties of function trend control

Introduction

You can use the TIA Portal Openness to get and set properties of function trend control in control screen items.

The following function trend control properties are supported in control screen items.

Property Name	Property Type	Accessibility
ShowStatisticRulers	bool	False
AreaSpacing	ushort	
Online	bool	
ExtendRulerToAxis	bool	
Caption	MultilingualProperty	False
WindowFlags	HmiWindowFlag	False
Icon	string	False
Top	Int	False
Left	int	False
Width	uint	False
Height	uint	False
CurrentQuality	HmiQuality	True
RequireExplicitRelease	bool	False
Authorization	string	False
Name	string	False
Visible	bool	False
Enabled	bool	False
TabIndex	ushort	False
ToolBar	HmiToolBarPart	True
StatusBar	HmiStatusBarPart	True
Font	HmiFontPart	True
TimeZone	int	False
ShowRuler	bool	False
ShiftAxes	bool	False
BackColor	Color	False
Legend	HmiLegendPart	True
EventHandlers	HmiFunctionTrendControl EventHandlerComposition	True
FunctionTrendAreas	HmiFunctionTrendAreaPartComposition	True

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Screen and screenitem is created
See Working with control screen items (Page 369)

Program code

Modify the following program code to access basic properties of function trend control:

```
//Width
var autoplay = hmifunctiontrendcontrol.Width;
hmifunctiontrendcontrol.Width = 1;
//BackColor
var backcolor = hmifunctiontrendcontrol.BackColor;
hmifunctiontrendcontrol.BackColor = Color.Beige;
//Name
var name = hmifunctiontrendcontrol.Name;
hmifunctiontrendcontrol.Name = "DefaultName";
```

Modify the following program code to access multilingual property access:

```
//Caption
var caption = hmifunctiontrendcontrol.Caption.Items[0].Text;
hmifunctiontrendcontrol.Caption.Items[0].Text =
"<body><p>TestforMultilingualProperty</p></body>";
```

Modify the following program code to access other typical properties of function trend control:

```
//Status bar
var statusBar = hmifunctiontrendcontrol.StatusBar;
var backColor = statusBar.BackColor;
statusBar.BackColor = Color.Aqua;
var enabled = statusBar.Enabled;
statusBar.Enabled = true;
var visible = statusBar.Visible;
statusBar.Visible = true;
```

Program code: TrendArea in FunctionTrendControl using part

Modify the following program code to create a trendarea in a HmiFunctionTrendControl using parts:

```
HmiSoftware hmiSoftware = GetHmiSoftware();
var screen = hmiSoftware.Screens;
var createdscreen = screen.Create("TestScreen_1");
HmiFunctionTrendControl funtnrend =
hmiSoftware.Screens[0].ScreenItems.Create<HmiFunctionTrendControl>("CTrendControl_1");
HmiFunctionTrendAreaPart part = funtnrend.FunctionTrendAreas.Create("part1");
```

Modify the following program code to searching a trendarea in HmiFunctionTrendControl using part:

```
HmiFunctionTrendAreaPart part = funtnrend.FunctionTrendAreas.Find("part1");
```

Modify the following program code to checking a trendarea in HmiFunctionTrendControl using part:

```
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiFunctionTrendControl funtnrend =
hmiSoftware.Screens[0].ScreenItems.Create<HmiFunctionTrendControl>("CTrendControl_1");
HmiFunctionTrendAreaPart part = funtnrend.FunctionTrendAreas.Create("part1");
bool bPresent = funtnrend.FunctionTrendAreas.Contains(part);
```

See also

- Connecting to the TIA Portal (Page 76)
- Opening a project (Page 106)
- Working with control screen items (Page 369)
- HMI Unified Software object (Page 272)

7.18.9.5 Faceplate instance screen items

Working with faceplate instance in control screen items

Introduction

You can perform the following tasks with faceplate instance items while using TIA Portal Openness:

- Creating faceplate instance (controls)
- Deleting faceplate instance (controls)
- Enumerating faceplate instance (controls)

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)
- Plant object type should be created

Program code

You can modify and use the following program code example while working with faceplate instance.

Creating faceplate instance

Modify the following program code to create faceplate instance screen item:

```
HmiSoftware hmiSoftware = GetHMISoftware ();
HmiScreen hmiScreen= hmiSoftware.Screens.Create ("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens [0].ScreenItems;
HmiFaceplateContainer faceplate = hmiScreen.ScreenItems.Create<HmiFaceplateContainer>
("abcd");
```

Recoverable Exception will be raised if the Create method is called with Screen Name with which Screen already exists in TIA Portal.

Deleting faceplate instance

Modify the following program code to delete faceplate instance screen item:

```
//Case 1
HmiSoftware hmiSoftware = GetHMISoftware ();
HmiScreen hmiScreen = hmiSoftware.Screens.Create ("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens [0].ScreenItems;
HmiFaceplateContainer faceplate = hmiScreen .ScreenItems.Create<HmiFaceplateContainer>
("abcd");
If (faceplate! = null)
{
faceplate.Delete ();
}
//Case 2
HmiSoftware hmiSoftware = GetHMISoftware ();
HmiScreen hmiScreen = hmiSoftware.Screens.Create ("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens [0].ScreenItems;
HmiFaceplateContainer faceplate = hmiScreen .ScreenItems.Create<HmiFaceplateContainer>
("abcd");
IEngineeringObject ObjhmiLineEnggObj = faceplate;
if (ObjhmiLineEnggObj != null)
{
ObjhmiLineEnggObj.Invoke ("Delete", null);
}
```

Enumerating faceplate instance

Modify the following program code to enumerate faceplate instance screen items of screen:

```
HmiSoftware hmiSoftware = GetHMISoftware ();
HmiScreen hmiScreen = hmiSoftware.Screens.Create ("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens [0].ScreenItems;
foreach (var item in screenitems)
{
//work with screenitems
}
```

Modify the following program code to searching faceplate instance from screen items list on the basis of name:

```
HmiScreenItemBase screenitems = hmiSoftware.Screens [0].ScreenItems.Find ("ScreenItems_1");
```

Modify the following program code to get faceplate instance from screenitems list on basis of index:

```
HmiScreenItemBase screenitem = hmiSoftware.Screens [0].ScreenItems [0];
```

Modify the following program code to checking a particular faceplate instance exist in screen item list by using Contains ():

```
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
HmiFaceplateContainer faceplate = screenitems .Create<HmiFaceplateContainer> ("abcd");
bool isexists = screenitems.Contains(faceplate);
```

See also

- [Connecting to the TIA Portal \(Page 76\)](#)
- [HMI Unified Software object \(Page 272\)](#)
- [Opening a project \(Page 106\)](#)

Accessing properties of faceplate instance

Introduction

You can use the TIA Portal Openness to get and set properties of faceplate instance in control screen items.

The following faceplate instance properties are supported in control screen items.

Property Name	Datatype	Description	Accessibility
Name	System.String	Name of the faceplate instance	R/W
ContainedType	System.String	Specifies contained faceplate type	R/W
Authorization	System.String	Authorization information for the faceplate instance	R/W
Enabled	System.Boolean	Allow operator control	R/W
CurrentQuality	Siemens.Engineering.HmiUnified.UI.Enum.Hmi-Quality	Connection Status	R
Height	System.UInt32	Specifies height of the control window	R/W
Icon	System.String	specifies the icon on the control window	R/W
Caption	Siemens.Engineering.MultilingualText	Text to be shown in the caption of a screen window or windowed control (Label)	R
RequireExplicitRelease	System.Boolean	If set to true, the screen item configured within the screen (or a parent screen if not configured locally) enables the screen item only when the release button while the button is pressed	R/W
TabIndex	System.UInt16	Screen items specifying a tab index of 0 are not part of the tab order	R/W

7.18 Functions for accessing the data of an HMI device (Unified)

Property Name	Datatype	Description	Accessibility
Visible	System.Boolean	Specifies the visibility of screen item	R/W
Width	System.UInt32	Specifies the width of the control window	R/W
WindowFlags	Siemens.Engineering.miUnified.UI.Enum.WindowFlag	Specifies the window configuration like ShowCaption, ShowBorder, AlwaysOnTop.	R/W
Top	System.Int32	Specifies the value of the Y coordinates of control window	R/W
Left	System.Int32	Specifies the value of the X coordinates of control window	R/W
Interface	Siemens.Engineering.HmiUnified.UI.Parts.HmiFaceplateInterfaceComposition	Faceplate interfaces	R

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)

Program code

Modify the following program code to get properties usage of faceplate instance:

```
HmiScreen screen = m_hmiSoftware.Screens[0];
HmiFaceplateContainer faceplate = screens[0].ScreenItems[0] as HmiFaceplateContainer;
//Get_Caption
string Caption = faceplate.Caption.Items[0].Text;
Console.WriteLine("Faceplate Caption: " + Caption);
//Get_WindowFlags
HmiWindowFlag WindowFlags = faceplate.WindowFlags;
Console.WriteLine("Faceplate WindowFlags: " + WindowFlags);
//Get_Icon
string Icon = faceplate.Icon;
Console.WriteLine("Faceplate Icon: " + Icon);
//Get_Top
int Top = faceplate.Top;
Console.WriteLine("Faceplate Top: " + Top);
//Get_Left
int Left = faceplate.Left;
Console.WriteLine("Faceplate Left: " + Left);
//Get_Width
uint Width = faceplate.Width;
Console.WriteLine("Faceplate Width: " + Width);
//Get_Height
uint Height = faceplate.Height;
Console.WriteLine("Faceplate Height: " + Height);
//Get_CurrentQuality
HmiQuality CurrentQuality = faceplate.CurrentQuality;
Console.WriteLine("Faceplate CurrentQuality: " + CurrentQuality);
//Get_RequireExplicitRelease
bool RequireExplicitRelease = faceplate.RequireExplicitRelease;
Console.WriteLine("Faceplate RequireExplicitRelease: " + RequireExplicitRelease);
//Get_Authorization
string Authorization = faceplate.Authorization;
Console.WriteLine("Faceplate Authorization: " + Authorization);
//Get_Name
string Name = faceplate.Name;
Console.WriteLine("Faceplate Name : " + Name);
//Get_Visible
bool Visible = faceplate.Visible;
Console.WriteLine("Faceplate Visible: " + Visible);
//Get_Enabled
bool Enabled = faceplate.Enabled;
Console.WriteLine("Faceplate Enabled: " + Enabled);
//Get_TabIndex
ushort TabIndex = faceplate.TabIndex;
Console.WriteLine("Faceplate TabIndex: " + TabIndex);
```

Modify the following programme code to set properties usage of faceplate instance:

```
//Set_Caption:  
faceplate.Caption.Items[0].Text = "testCaption";  
//Set_WindowFlags  
faceplate.WindowFlags = HmiWindowFlag.CanSize;  
//Set_Icon  
faceplate.Icon = "testCaption";  
//Set_Top  
faceplate.Top = 50;  
//Set_Left  
faceplate.Left = 50;  
//Set_Width  
faceplate.Width = 50;  
//Set_Height  
faceplate.Height = 50;  
//Set_RequireExplicitRelease  
faceplate.RequireExplicitRelease = true;  
//Set_Authorization  
faceplate.Authorization = Authorization;  
//Set_Name  
faceplate.Name = "TestName";  
//Set_Visible  
faceplate.Visible = true;  
//Set_Enabled  
faceplate.Enabled = true;  
//Set_TabIndex  
faceplate.TabIndex = 10;
```

Note

Get - Recoverable exception will be thrown in case respective consistency check fails for the specified property. During setting of properties, the set operation fails consistency checks then a Recoverable exception would be raised.

See also

- Connecting to the TIA Portal (Page 76)
- Opening a project (Page 106)
- HMI Unified Software object (Page 272)
- Working with control screen items (Page 369)

Working with dynamization and events of faceplate instance

Introduction

You can perform the following tasks related to dynamizations and event objects of faceplate properties using TIA Portal Openness:

- Accessing dynamization (tag connections, animations, local scripts) for faceplate instances
- Accessing faceplate instance events with scripts and system functions
- Accessing property events of faceplate instance

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)

Program code: Accessing dynamization for faceplate instances

For description on Accessing dynamization for faceplate instances, see Working with dynamization for screen/ screen items (Page 404).

Program code: Accessing faceplate instance events with scripts and system functions

For description on Accessing faceplate instance events with scripts and system functions, see Working with dynamization & events for screen/screen items using scripts (Page 401)

Program code: Accessing property events for faceplate instance

For description on Accessing property events for faceplate instance, see Working with events for screen/screen items & properties (Page 398)

See also

- Connecting to the TIA Portal (Page 76)
- Opening a project (Page 106)
- HMI Unified Software object (Page 272)
- Working with dynamization for screens/screen items (Page 404)
- Working with dynamization & events for screen/screen items using scripts (Page 401)
- Working with events for screen/screen items & properties (Page 398)

7.18.9.6 Working with events for screen/screen items & properties

Introduction

You can perform the followings tasks with property events and events for screen and screen items while using TIA Portal Openness:

- Creating property events
- Enumerating property events
- Deleting property events
- Accessing properties of events
- Creating events for screens and screen items
- Enumerating events for screens and screen items
- Accessing properties of events for screen and screen items
- Deleting events for screen and screen items

Requirement

- The TIA Portal Openness is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Unified Software object
See HMI Unified Software object (Page 272)

Program code

You can modify and use the following program code examples while working with events for screen/screen items.

Creating property events

Modify the following program code to create property events:

```
HmiScreen screen = ((HmiSoftware)targetSW).Screens.Create("MyScreen");
HmiCircle hmiCircle = screen.ScreenItems.Create<HmiCircle>("MYCircle");
//Event Creation
PropertyEventHandler propertyEvent = hmiCircle.PropertyEventHandlers.Create("ToolTipText",
PropertyEventType.Change);
```

Enumerating property events

Modify the following program code to enumerate property events:

```
HmiScreen screen = ((HmiSoftware)targetSW).Screens.Create("MyScreen");
HmiCircle hmiCircle = screen.ScreenItems.Create<HmiCircle>("MYCircle");
var hmiScreenPropeventDyn = screen.PropertyEventHandlers.Create("Enabled",
PropertyEventType.Change);
var enabledEvent = screen.PropertyEventHandlers.Find("Enabled", PropertyEventType.Change);
//Event Browse
Console.WriteLine("CountofEventActions"+ screen.PropertyEventHandlers.Count.ToString());
```

Deleting property events

Modify and use the following program code to delete property events:

```
public void Delete()
PropertyEventHandler propertyEvent =
((HmiSoftware)targetSW).Screens[0].ScreenItems[0].PropertyEventHandlers.Create("ToolTipText",
PropertyEventType.Change);
var changedEvent = screen.PropertyEventHandlers.Find("ToolTipText",
PropertyEventType.Change);
//Event Deletion
changedEvent.Delete();
```

Accessing properties of events

Modify the following program code to get event properties:

```
//Event Get
PropertyEventType eventType = propertyEvent.EventType;
string propertyName = propertyEvent.PropertyName;
IHmiScript hmiScript = propertyEvent.Script;
//Script Action in Event Get
bool hmiScriptEventAsync = hmiScript.Async;
string hmiScriptEventGlobal = hmiScript.GlobalDefinitionAreaScriptCode;
string hmiScriptEventCode = hmiScript.ScriptCode;
HmiValidationResult result = hmiScript.SyntaxCheck();
```

Creating events for screen and screen items

Modify the following program code to create the event at the screen or screen item level object using EventHandler navigator:

```
var screenComp = ((HmiSoftware)targetSWTag).Screens;
HmiScreen screenOne = screenComp.Create("Screen2");
HmiScreenEventHandler screenHandler =
screenOne.EventHandlers.Create(HmiScreenEventType.Unloaded);
```

Enumerating events for screen and screen items:

Modify the following program code to enumerate events for screen and screen items:

```
public EventHandler Find(string propertyName);
var screenComp = ((HmiSoftware)targetSWTag).Screens;
HmiScreen screenOne = screenComp.Create("Screen2");
HmiScreenEventHandler screenHandler =
screenOne.EventHandlers.Create(HmiScreenEventType.Unloaded);
HmiScreenEventHandler screenHandlerSearched =
screenOne.EventHandlers.Find(HmiScreenEventType.Unloaded);
```

Accessing properties of events for screen and screen items:

Modify the following program code to get properties of events for screen and screen items:

```
var screenComp = ((HmiSoftware)targetSWTag).Screens;
HmiScreen screenOne = screenComp.Create("Screen2");
HmiScreenEventHandler screenHandler =
screenOne.EventHandlers.Create(HmiScreenEventType.Unloaded);
Console.WriteLine(screenOne.Name + ".EventHandler.EventType - {0}",
screenHandler.EventType);
Console.WriteLine(screenOne.Name + ".EventHandler.Script.Async - {0}",
screenHandler.Script.Async);
Console.WriteLine(screenOne.Name + ".EventHandler.Script.GlobalDefinitionAreaScriptCode - {0}",
screenHandler.Script.GlobalDefinitionAreaScriptCode);
Console.WriteLine(screenOne.Name + ".EventHandler.Script.ScriptCode - {0}",
screenHandler.Script.ScriptCode);
```

Deleting events for screen and screen items:

Modify the following program code to delete events for screen and screen items:

```
public void Delete ()
HmiSoftware hmisoftware = GetHMISoftware();
HmiScreen hmiscreen = hmisoftware.Screens.Create(Guid.NewGuid().ToString());
var hmiscreenPropEventDyn = hmiscreen.PropertyEventHandlers.Create("Enabled",
PropertyEventType.Change);

var enabledevent = hmiscreen.PropertyEventHandlers.Find("Enabled",
PropertyEventType.Change);
enabledevent.Delete();
```

Note

Create - Recoverable exception will be thrown in case respective property events and events for screen and screen items is not supported for the specified property. If during set of properties the set operation fails consistency checks then a Recoverable exception would be raised.

See also

- Connecting to the TIA Portal (Page 76)
- Opening a project (Page 106)
- HMI Unified Software object (Page 272)

7.18.9.7 Working with dynamization & events for screen/screen items using scripts

Introduction

You can perform the following tasks with script to configure property dynamization and events for screen/screen items while using TIA Portal Openness:

- Creating script dynamizations
- Enumerating script dynamizations
- Deleting script dynamizations
- Accessing properties of script dynamizations
- Accessing properties of trigger
- Checking syntax

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Unified Software object
See HMI Unified Software object (Page 272)

Program code

You can modify and use the following program code examples while working with script dynamization for screen/screen items.

Creating script dynamizations

Modify the following program code to create script dynamizations for a property:

```
public DynamizationBase Create<DynamizationBase>(string propertyName);  
HmiScreen screen = m_hmisoftware.Screen[0];  
DynamizationBaseComposition dyns = screen.Dynamizations;  
ScriptDynamization scriptDynamic = dyns.Create<ScriptDynamization>("BackColor");
```

Enumerating script dynamizations

Modify the following program code to enumerate script dynamization of a property:

```
public DynamizationBase Find(string propertyName);
HmiScreen screen = m_hmiSoftware.Screens[0];
DynamizationBaseComposition dysns = screen.Dynamizations;
if (dysns is ScriptDynamization)
{
    ScriptDynamization scriptDynamic = (ScriptDynamization)dysns.Find("BackColor");
}
```

Deleting script dynamizations

Modify the following program code to delete script dynamization for a property:

```
HmiScreen screen = m_hmiSoftware.Screens[0];
DynamizationBaseComposition dysns = screen.Dynamizations;
if (dysns is ScriptDynamization)
    ScriptDynamization scriptDynamic = (ScriptDynamization)dysns.Find("BackColor");
    scriptDynamic.Delete();
```

Accessing properties of script dynamizations

Modify the following program code to get and set properties of script dynamization:

```
HmiScreen screen = m_hmiSoftware.Screens[0];
DynamizationBaseComposition dysns = screen.Dynamizations;
foreach (DynamizationBase dynamic in dysns)
{
    if (dynamic.DynamizationType == DynamizationType.Script)
    {
        ScriptDynamization scriptDynamization = (ScriptDynamization)dynamic;
        scriptDynamization.Async = true;
        scriptDynamization.GlobalDefinitionAreaScriptCode = @"Add(parameter1,parameter2)";
        scriptDynamization.ScriptCode = @"
var value;
let tag1 = Tags('Tag_1');
let tagValue1 = tag1.Read();
HMIRuntime.Trace('value of MyTag1: ' + tagValue1);
return value; ";
        Trigger triggerObj = scriptDynamization.Trigger;
    }
}
```

Accessing properties of Trigger

Modify the following program code to get and set properties of trigger:

```
HmiScreen screen = m_hmiSoftware.Screens[0];
DynamizationBaseComposition dyns = screen.Dynamizations;
foreach (DynamizationBase dynamic in dyns)
{
if (dynamic.DynamizationType == DynamizationType.Script)
{
ScriptDynamization scriptDynamization = (ScriptDynamization)dynamic;
// Property write
scriptDynamization.Trigger.Type = TriggerType.CustomCycle;
scriptDynamization.Trigger.CustomDuration = "T500ms";
scriptDynamization.Trigger.Tags = new List<string>() { "Tag_1" };
// Property read
Console.WriteLine(scriptDynamization.Trigger.Type);
Console.WriteLine(scriptDynamization.Trigger.CustomDuration);
Console.WriteLine(scriptDynamization.Trigger.Tags);
}
}
```

Syntax Check

Modify the following program code to check syntax for script code or global script code using SyntaxCheck action (method) of script dynamization:

```
HmiScreen screen = m_hmiSoftware.Screens[0];
DynamizationBaseComposition dyns = screen.Dynamizations;
foreach (DynamizationBase dynamic in dyns)
{
if (dynamic.DynamizationType == DynamizationType.Script)
{
ScriptDynamization scriptDynamization = (ScriptDynamization)dynamic;
scriptDynamization.ScriptCode = @""
var value;let tag1 = Tags('Tag_1');
let tagValue1 = tag1.Read();
HMIRuntime.Trace('value of MyTag1: ' + tagValue1);
return value; ";
HmiValidationResult syntaxCkResult = scriptDynamization.SyntaxCheck();
IEnumerable<string> error = syntaxCkResult.Errors;
IEnumerable<string> warnings = syntaxCkResult.Warning;
}
}
```

Note

Create - Recoverable exception will be thrown in case respective dynamization is not supported for the specified property. If during set of properties the set operation fails consistency checks then a Recoverable exception would be raised.

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

[HMI Unified Software object \(Page 272\)](#)

7.18.9.8 Working with dynamization for screens/screen items

Introduction

You can perform the following tasks with dynamization feature of screens/screen items while using TIA Portal Openness:

- Creating dynamization for a property
- Enumerating dynamization for a property
- Deleting dynamization for a property
- Accessing common properties of dynamization
- Accessing properties of tag dynamization
- Accessing properties of flasing dynamization
- Accessing properties of resourcelist dynamization

Note

For accessing dynamization for a properties at part level, you need to access respective part of Control. For part information, see [Accessing properties of trend control \(Page 76\)](#)

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See [Connecting to the TIA Portal \(Page 76\)](#)
- A Project is open
See [Opening a project \(Page 106\)](#)
- Access to HMI Software object
See [HMI Unified Software object \(Page 272\)](#)

Program code

You can modify and use the following program code examples while working with dynamizations for screen/screen items.

Creating dynamization for a property

Modify the following program code to create dynamization for a property:

```
HmiScreen screen = m_hmiSoftware.Screens[0];
DynamizationBaseComposition dysns = screen.Dynamizations;
// Flashing dynamization
FlashingDynamization FlashingDyn = dysns.Create<FlashingDynamization>("BackColor");
// ResourceList dynamization
ResourceListDynamization ResListDyn = dysns.Create<ResourceListDynamization>("DisplayName");
// Tag dynamization
TagDynamization tagDyn = dysns.Create<TagDynamization>("Width");
```

Enumerating dynamization of a property

Modify the following program code to enumerate dynamization for a property:

```
HmiScreen screen = m_hmiSoftware.Screens[0];
DynamizationBaseComposition dysns = screen.Dynamizations;
// Flashing dynamization
FlashingDynamization FlashDyn = (FlashingDynamization)dysns.Find("BackColor");
// ResourceList dynamization
ResourceListDynamization resourceDyn = (ResourceListDynamization)dysns.Find("DisplayName");
// Tag dynamization
TagDynamization tagDyn = (TagDynamization)dysns.Find("Width");
```

Deleting dynamization of a property

Modify the following program code to delete dynamization of a property:

```
public void Delete ()
HmiScreen screen = m_hmiSoftware.Screens[0];
DynamizationBaseComposition dysns = screen.Dynamizations;
DynamizationBase dynamization = dysns.Find("BackColor");
dynamization.Delete();
```

Accessing common properties of dynamization

Modify the following program code to get common properties of dynamization:

```
HmiSoftware m_hmiSoftware = ...;
DynamizationBaseComposition screenDynamizations = m_hmiSoftware.Screens[0].Dynamizations;
DynamizationBase screenDynamization = screenDynamizations.Find("BackColor");
string propertyName = screenDynamization.PropertyName;
DynamizationType dynamizationType = screenDynamization.DynamizationType;
```

Accessing properties of tag dynamization

Modify the following program code to get and set properties of tag dynamization:

```
HmiScreen screen = m_hmiSoftware.Screens[0];
TagDynamization tagDynamization = (TagDynamization)screen.Dynamizations.Find("Width");
foreach (DynamizationBase dynamization in screen.Dynamizations)
{
if (dynamization.DynamizationType == DynamizationType.Tag)
{
TagDynamization TagDynamization = (TagDynamization)dynamization;
TagDynamization.Tag = "HmiTagName";
TagDynamization.UseIndirectAddressing = true;
TagDynamization.ReadOnly = true;
}
}
```

Accessing properties of flashing dynamization

Modify the following program to get and set properties of flashing dynamization:

```
HmiScreen screen = m_hmiSoftware.Screens[0];
foreach (DynamizationBase dynamization in screen.Dynamizations)
{
if (dynamization.DynamizationType == DynamizationType.Floating)
{
FlashingDynamization floatingDynamization = (FlashingDynamization)dynamization;
floatingDynamization.AlternateColor = Color.Beige;
floatingDynamization.Color = Color.FromArgb(12, 22, 152, 200);
floatingDynamization.FloatingCondition = FloatingCondition.Always;
floatingDynamization.FloatingRate = FlashingRate.Medium;
}
}
```

Accessing properties of resourcelist dynamization

Modify the following program code to get and set properties of resourcelist dynamization

```
HmiScreen screen = m_hmiSoftware.Screens[0];
foreach (DynamizationBase dynamization in screen.Dynamizations)
{
if (dynamization.DynamizationType == DynamizationType.ResourceList)
{
ResourceListDynamization resourceListDyn = (ResourceListDynamization)dynamization;
resourceListDyn.Tag = "Tag name";
resourceListDyn.ResourceList = "Resource list name";
}
}
```

Note

Create - Recoverable exception will be thrown in case respective dynamization is not supported for the specified property. If during set of properties the set operation fails consistency checks then a Recoverable exception would be raised.

See also

- Connecting to the TIA Portal (Page 76)
- Opening a project (Page 106)
- HMI Unified Software object (Page 272)
- Accessing properties of trend control (Page 382)

7.18.9.9 Checking license for access Unified device

Introduction

You can use the TIA Openness interface to check any Openness client application for the presence of WinCC Unified license, so that a client application cannot bypass licensing using Openness interfaces.

You are only able to access to an HMI Unified device and all its underlying objects (e.g. tags, alarms, screen, etc.) when HMI Unified license is available. If WinCC Unified is not licensed than any operation on device level (e.g. create, find, delete) results in a null value being returned from Openness API.

Note

For HMI Unified Software, If license is invalid, then a null value will be returned.

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Access to HMI Software object
See HMI Unified Software object (Page 272)

Program code

Modify the following program code to access the device or device sub systems when license is not available, a null value is returned:

```
private static void ListDevices(Project project)
{
foreach (var device in project.Devices)
{
ListDeviceItem(device);
}
}
private static void ListDeviceItem(Device device)
{
Console.WriteLine("HMI device - " + device.Name);
foreach (var item in device.DeviceItems)
{
Console.WriteLine("HMI device type - " + item.TypeIdentifier);
var softContainer = item.GetService<SoftwareContainer>();
if (softContainer != null)
{
var softTarget = softContainer.Software;
if (softTarget != null)
{
Console.WriteLine("HMI device software - " + softTarget.Name);
}
}
}
}
```

Note

While accessing the above program code, the softTarget will be null, and accessing sub-systems, such as alarms, tags will not be possible.

Modify the following program code to access the device or device sub systems when license is available, a null value is returned.

```
private static void ListDevices(Project project)
{
foreach (var device in project.Devices)
{
ListDeviceItem(device);
}
}
private static void ListDeviceItem(Device device)
{
Console.WriteLine("HMI device - " + device.Name);
foreach (var item in device.DeviceItems)
{
Console.WriteLine("HMI device type - " + item.TypeIdentifier);
var softContainer = item.GetService<SoftwareContainer>();
if (softContainer != null)
{
var softTarget = softContainer.Software;
}
if (softTarget != null)
{
Console.WriteLine("HMI device software - " + softTarget.Name);
}
}
}
```

Note

While accessing the above program code, the softTarget will have device container, and accessing sub-systems, such as alarms, tags will be possible.

The return value of NULL for device container denotes that the license is not available. The recoverable exception does not support custom exception messages, hence the use of NULL as a return value.

See also

- [Connecting to the TIA Portal \(Page 76\)](#)
- [HMI Unified Software object \(Page 272\)](#)

7.18.10 Accessing common plant model hierarchies

7.18.10.1 Working with plant view

Introduction

You can perform the following tasks related to plant view while using TIA Portal Openness:

- Create plant view
- Delete plant view
- Rename plant view

Requirement

- The TIA Portal Openness is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)

Program code

Modify the following program code to create a plant view:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;  
PlantView plantView = plantViews.Create("ABCManufacturingPlant");
```

Note

You can create only one plant view inside a project.

Modify the following program code to delete or remove a plant view:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;  
PlantView plantView = plantViews.Find("ABCManufacturingPlant");  
plantView.Delete();
```

Modify the following program code to rename a plant view by updating the 'Name' property:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;  
PlantView plantView = plantViews.Find("ABCManufacturingPlant");  
plantView.Name = "New_PlantView_Name";
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.18.10.2 Working with plant view nodes

Introduction

You can perform the following tasks related to plant view node while using TIA Portal Openness:

- Create plant view nodes
- Delete plant view nodes
- Rename plant view nodes

Properties

The following properties are supported in Plant View Node using TIA Portal Openness:

Property Name	Data Type	Accessibility
HierarchyPath	String	R
PlantObject	Siemens.Engineering.HmiUnified.Cpm.PlantObject	R
Name	String	R/W
PlantView	String	R
ViewPath	String	R
PlantViewNodes	PlantViewNodeComposition	R

Requirement

- The TIA Portal Openness is connected to the TIA Portal
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open
See [Opening a project \(Page 106\)](#)

Program code

Modify the following program code to create plant view node under a plant view:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Find("ABCManufacturingPlant");
PlantViewNodeComposition viewNodes = plantView.PlantViewNodes;
PlantViewNode mixingNode = viewNodes.Create("Mixing");
```

Modify the following program code to create plant view node under another plant view node:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Find("ABCManufacturingPlant");
PlantViewNodeComposition viewNodes = plantView.PlantViewNodes;
PlantViewNode mixingNode = viewNodes.Create("Mixing");
PlantViewNode motorNode = mixingNode.Create("Motor");
```

Note

You can create any number of plant view nodes inside a plant view or plant view node itself.

Modify the following program code to remove or delete a plant view node:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Find("ABCManufacturingPlant");
PlantViewNode mixingNode = plantView.PlantViewNodes.Find("Mixing");
mixingNode.Delete();
```

Modify the following program code to rename a plant view node by updating the 'Name' property:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Find("ABCManufacturingPlant");
PlantViewNodeComposition viewNodes = plantView.PlantViewNodes;
PlantViewNode mixingNode = viewNodes.Create("Mixing");
mixingNode.Name = "New_PlantViewNode_Name";
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.18.10.3 Enumerating plant view

Introduction

You can use the TIA Portal Openness to enumerate plant view present on given project in TIA Portal.

Requirement

- The TIA Portal Openness is connected to the TIA Portal
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open
See [Opening a project \(Page 106\)](#)

Program code

Modify the following program code to enumerate all plant views of the project:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
foreach (var item in plantViews)
{
// work with plant views (currently only 1 plant view can be created inside a project)
}
```

Modify the following program code to find a plant view from plant view list based on the name:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Find("ABCManufacturingPlant");
```

Modify the following program code to find a plant view from plant view list based on index:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews[0];
```

Modify the following program code to find plant view node from the plant view list based on the hierarchy path of the required plant view node:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantViewNode plantViewNode = plantViews.FindPlantViewNode("ABCManufacturingPlan\Mixing
\Motor");
```

Modify the following program code to find a plant view node from the plant view node list based on the name of the plant view:

```
PlantViewNodeComposition plantViewNodes =
m_tiaPortalApp.Projects[0].PlantViews[0].PlantViewNodes;
PlantViewNode plantViewNode = plantViewNodes.Find("Mixing");
```

Modify the following program code to enumerate all plant view nodes list available inside plant view:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantViewNodeComposition nodes = plantViews.PlanViewNodes;
foreach (var item in nodes)
{
// work with plant view nodes inside plant view
}
```

7.18 Functions for accessing the data of an HMI device (Unified)

Modify the following program code to enumerate all plant view nodes available inside any hierachial level:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantViewNodeComposition nodes = plantView.PlantViewNodes;
PlantViewNodeComposition subNodes = nodes[2].PlantViewNodes;
foreach (var item in subNodes)
{
// work with plant view nodes inside 2nd node within the plant view
}
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.18.10.4 Working with plant view to device

Introduction

You can perform the following tasks related to accessing plant view hierarchies and plant object while using TIA Portal Openness:

- Assign device to the plant view
- Change device to the plant view
- Remove device to the plant view

Requirement

- The TIA Portal Openness is connected to the TIA Portal
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open
See [Opening a project \(Page 106\)](#)

Program code

Modify the following program code to assign a device to the plant view by setting the AssignedHmiDevice property:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Create("ABCManufacturingPlant");
plantView.AssignedHmiDevice = "HMI_RT_1";
```

Modify the following program code to change a device to the plant view by setting a device name to AssignedHmiDevice property.

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;  
PlantView plantView = plantViews.Create("ABCManufacturingPlant");  
plantView.AssignedHmiDevice = "HMI_RT_2";
```

Modify the following program code to remove device to the plant view by setting a blank device name to AssignedHmiDevice property.

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;  
PlantView plantView = plantViews.Create("ABCManufacturingPlant");  
plantView.AssignedHmiDevice = string.Empty;
```

Note

To remove the device, use only string.empty. Passing NULL or whitespace (" "), will be treated as a device name and will return device not found error message

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.18.11 Accessing common plant model instance**7.18.11.1 Working with CPM object instance****Introduction**

You can perform the following tasks related to access CPM instance based on CPM type while using TIA Portal Openness:

- Create CPM object instances
- Delete CPM object instances
- Rename CPM object instances

Requirement

- The TIA Portal Openness is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- Create a Plant

Program code

Modify the following program code to create CPM object instance:

```
// Instance inside plant view
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Create("ABCManufacturingPlant");
plantView.AssignedHmiDevice = "HMI_RT_1";
PlantViewNodeComposition nodes = plantView.PlantViewNodes;
PlantViewNode node = nodes.Create("ObjectLevel1", "Plant_Object_Type_1");
//Instance inside plant view node
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Create("IngredientMixer",
"Plant_Object_Type_1");
```

Modify the following program code to remove or delete a CPM object instance:

```
PlantViewNode mixingNode = plantView.PlantViewNodes.Find("Mixing");
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Find("IngredientMixer");
mixerObject.Delete();
```

Modify the following program code to rename a CPM object instance by updating the 'Name' property:

```
PlantViewNode mixingNode = plantView.PlantViewNodes.Find("Mixing");
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Find("IngredientMixer");
mixerObject.Name = "New_PlantViewNode_Name";
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.18.11.2 Working with plant view nodes of CPM object instance

Introduction

You can perform the following tasks related to access plant view nodes of CPM object instances while using TIA Portal Openness:

- Create plant view nodes
- Delete plant view nodes
- Rename plant view nodes

Requirement

- The TIA Portal Openness is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)

Program code

Modify the following program code to create a plant view node inside a CPM object instance

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Find("ABCManufacturingPlant");
PlantViewNode mixerObject = plantView.PlantViewNodes.Create("ObjectLevel1",
"Plant_Object_Type_1");
PlantViewNode mixingNode = mixerObject.PlantViewNodes.Create("Mixing");
```

Note

You can create any number of plant view node inside a CPM object

Modify the following program code to create a CPM object instance inside a plant view node:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Find("ABCManufacturingPlant");
PlantViewNode mixingNode = plantView.PlantViewNodes.Create("Mixing");
PlantViewNode mixerObject = mixingNode.PlantViewNodes.Create("ObjectLevel1",
"Plant_Object_Type_1");
```

Note

You can create any number of CPM object instances inside a plant view node

7.18 Functions for accessing the data of an HMI device (Unified)

Modify the following program code to remove or delete a plant view node from a plant view node:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Find("ABCManufacturingPlant");
PlantViewNode mixerObject = plantView.PlantViewNodes.Find("ObjectLevel1");
PlantViewNode mixingNode = mixerObject.PlantViewNodes.Find("Mixing");
mixingNode.Delete();
```

Modify the following program code to rename a plant view node of CPM object instance by updating the 'Name' property:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Find("ABCManufacturingPlant");
PlantViewNode mixerObject = plantView.PlantViewNodes.Find("ObjectLevel1",
"Plant_Object_Type_1");
PlantViewNode mixingNode = mixerObject.PlantViewNodes.Find("Mixing");
mixingNode.Name = "New_PlantViewNode_Name";
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.18.11.3 Enumerating interface tags of CPM object instance

Introduction

You can use the TIA Portal Openness to enumerate a member of a CPM object instance.

Requirement

- The TIA Portal Openness is connected to the TIA Portal
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open
See [Opening a project \(Page 106\)](#)

Program code

Modify the following program code to enumerate all members of the CPM object instance:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews[0];
PlantViewNodeComposition plantViewNodes = plantView.PlantViewNodes;
PlantViewNode plantViewNode = plantViewNodes.Find("MixerObject_1");
PlantObject plantObject = plantViewNode.PlantObject;
PlantObjectInterfaceComposition interfaces = plantObject?.PlantObjectInterfaces;
```

Note

In the above example, the ?.operator refers to null check

Modify the following program code to find an interface tag from interfaces list based on 'Name':

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews[0];
PlantViewNodeComposition plantNodes = plantView.PlantViewNodes;
PlantViewNode plantViewNode = plantNodes.Find("MixerObject_1");
PlantObject plantObject = plantViewNode.PlantObject;
PlantObjectInterface interface = plantObject. PlantObjectInterfaces.Find("Interface_1");
```

Modify the following program code to find an interface tag from interface list based on index:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews[0];
PlantViewNodeComposition plantViewNodes = plantView.PlantViewNodes;
PlantViewNode plantViewNode = plantViewNodes.Find("MixerObject_1");
PlantObject plantObject = plantViewNode.PlantObject;
PlantObjectInterfaceComposition interfaces = plantObject.PlantObjectInterfaces;
PlantObjectInterface interface = interfaces[0];
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.18.12 Accessing properties for interface/logging tags of plant object instances

7.18.12.1 Accessing/Updating properties of interface tags of CPM plant object instances

Introduction

You can use the TIA Portal Openness to access and update the properties on interface tags of CPM plant object instance.

The following properties can be accessed on interface tags of CPM plant object instance:

Property Name	Data type	Description	Accessibility
Name	System.String	Name of interface tags	R/W
PlcTag	System.String	PLC tag associated with interface tags	R/W
Connection	System.String	Connection of interface tags	R/W

7.18 Functions for accessing the data of an HMI device (Unified)

Property Name	Data type	Description	Accessibility
PlcName	System.String	PLC name associated with interface tags	R
AccessMode	PlantObjectTagAccessMode	Access mode for interface tags	R
DataType	System.String	Object type of the interface tags	R
MaxLength	System.Int	Length of interface tags	R
HmiDataType	System.Int	HMI data type of interface tags	R
AcquisitionMode	PlantObjectTagAcquisitionMode	Acquisition mode of interface tags	R/W
AcquisitionCycle	System.Int	Acquisition cycle for interface tags	R/W
Persistent	System.Bool	Persistence for internal tags of CPM object instance	R
Comment	System.String	Comment	R/W
Members	PlantObjectInterfaceMemberComposition	Members of Interface Tag	R

Note

PLCTag & Connection property can only be configured after assigning HMI device to the plant view.

Requirement

- The TIA Portal Openness is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)

Program code

Modify the following program code to access all the properties of interface tags:

```
// Instance inside plant view node
PlantViewNode mixingNode = plantView.PlantViewNodes.Create("Mixing");
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Create("IngredientMixer", "Mixer");
PlantObjectInterfaceComposition interfaceTags =
mixerObject.PlantObject.PlantObjectInterfaces;
PlantObjectInterface interfaceTag = interfaceTags.FirstOrDefault();
string name = interfaceTag.Name;
MultilingualText comment = interfaceTag.Comment
```

Modify the following program code to update some properties of an interface tag, such as Comment, PLC Tag, Connection, Acquisition Mode, Acquisition Cycle:

```
// Instance inside plant view node
PlantViewNode mixingNode = plantView.PlantViewNodes.Create("Mixing");
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Create("IngredientMixer", "Mixer");
PlantObjectInterfaceComposition interfaceTags =
mixerObject.PlantObject.PlantObjectInterfaces;
PlantObjectInterface interfaceTag = interfaceTags.FirstOrDefault();
interfaceTag.Name = "New_InterfaceTag_Name";
interfaceTag.Comment.Items[0].Text = "New_InterfaceTag_Comment";
interfaceTag.AcquisitionMode = PlantObjectTagAcquisitionMode.CyclicContinuous;
interfaceTag.AcquisitionCycle = "T12s";
interfaceTag.Connection = "Connection_1";
interfaceTag.PlcTag = "PLC_Tag_1";
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.18.12.2 Accessing/Updating properties of members tags of interface tags

Introduction

You can use the TIA Portal Openness to access and update properties of member tags of interface tags.

The following properties can be accessed on member tags of interface tags:

Property Name	Data type	Description	Accessibility
Name	System.String	Name of member tags	R
DataType	System.String	Object type of the member tags	R
MaxLength	System.Int	Length of member tags	R
HmiDataType	System.String	HMI data type of member tags	R
InitialMinValue	PlantObjectTagLowerRange	Lower range value	R/W
InitialMaxValue	PlantObjectTagUpperRange	Upper range value	R/W
InitialValue	System.Object	Initial value	R/W
SubstituteValue	PlantObjectTagSubstituteValue	Substitute value	R
Comment	MultilingualText	Comment	R/W
Members	PlantObjectInterfaceMemberComposition	Members of Member Tag	R
LoggingTags	PlantObjectLoggingTagComposition	LoggingTags of Member Tag	R

Requirement

- The TIA Portal Openness is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)

Program code

Modify the following program code:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews[0];
PlantViewNodeComposition plantViewNodeComposition = plantView.PlantViewNodes;
PlantViewNode plantViewNode = plantViewNodeComposition[0];
PlantObjectInterfaceComposition tagComposition =
plantViewNode.PlantObject.PlantObjectInterfaces;
PlantObjectInterface interfaceTag = tagComposition[0];
PlantObjectInterfaceMemberComposition memberTags = interfaceTag.Members;
PlantObjectInterfaceMember memberTag = memberTags.FirstOrDefault();
//GetAttributes
List<string> listOfPropertyNames = new List<string>()
{
    "Name"
};
var listOfPropertyValues = memberTag.GetAttributes(listOfPropertyNames);
//SetAttributes
IEnumerable<KeyValuePair<string, object>> propertyNameValuePairList = new
List<KeyValuePair<string, object>>()
{
    new KeyValuePair<string, object>("Name", "MemberTag_1");
};
memberTag.SetAttributes(propertyNameValuePairList);
```

Modify the following program code to access all the properties of member tags:

```
// Instance inside plant view node
PlantViewNode mixingNode = plantView.PlantViewNodes.Create("Mixing");
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Create("IngredientMixer", "Mixer");
PlantObjectInterfaceComposition interfaceTags =
mixerObject.PlantObject.PlantObjectInterfaces;
PlantObjectInterfaceMemberComposition memberTags = interfaceTag.Members;
PlantObjectInterfaceMember memberTag = memberTags.FirstOrDefault();
string name = memberTag.Name;
string comment = memberTag.Comment.Items[0].Text;
```

Modify the following program code to update properties of a member tag of a CPM object instance:

```
// Instance inside plant view node
PlantViewNode mixingNode = plantView.PlantViewNodes.Create("Mixing");
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Create("IngredientMixer", "Mixer");
PlantObjectInterfaceComposition interfaceTags =
mixerObject.PlantObject.PlantObjectInterfaces;
PlantObjectInterfaceMember memberTag = memberTags.FirstOrDefault();
memberTag.Comment.Items[0].Text = "New_MemberTag_Comment";
memberTag.InitialValue = 2;
memberTag.InitialMinValue.Value = 2;
memberTag.InitialMinValue.ValueType = PlantObjectTagLimitValueType.Constant;
memberTag.InitialMaxValue.Value = 2;
memberTag.InitialMaxValue.ValueType = PlantObjectTagLimitValueType.Constant;
```

Modify the following code to access all the logging tags of a member tag of CPM object instance:

```
// Instance inside plant view node
PlantViewNode mixingNode = plantView.PlantViewNodes.Create("Mixing");
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Create("IngredientMixer", "Mixer");
PlantObjectInterfaceComposition interfaceTags =
mixerObject.PlantObject.PlantObjectInterfaces;
PlantObjectInterface interfaceTag = interfaceTags.FirstOrDefault();
PlantObjectInterfaceMemberComposition memberTags = interfaceTag.Members;
PlantObjectInterfaceMember memberTag = memberTags.FirstOrDefault();
PlantObjectLoggingTagComposition loggingTags = memberTag.LoggingTags;
PlantObjectLoggingTag loggingTag = loggingTags.FirstOrDefault();
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.18.12.3 Accessing/Updating properties of logging tags of member tags

Introduction

You can use the TIA Portal Openness to access logging tags of member tags present inside CPM object instance.

7.18 Functions for accessing the data of an HMI device (Unified)

The following properties can be accessed on logging tags of member tags present inside CPM object instance:

Property name	Data type	Description	Accessibility
AggregationDelay	System.TimeSpan	Aggregation delay value	R
AggregationMode	PlantObjectLoggingTagAggregationMode	Aggregation Mode value	R
Cycle	System.String	Logging Cycle of logging tag	R
CycleFactor	System.UInt32	Logging Cycle Factor vlaue	R
DataLog	System.String		R
Source	System.string	Source Logging Tag	R
TriggerMode	PlantObjectLoggingTagTriggerMode	TriggerMode of Logging tag	R
TriggerTag	System.String	TriggerTag Value	R
TriggerTagBitNumber	System.UInt32	TriggerTagBitNumber value	R
Name	System.string	Name of logging tag	R
LogConfiguration	System.String	Data log of logging tag	R
LoggingMode	PlantObjectLoggingTagLoggingMode	Logging mode of logging tag	R
SmoothingMode	PlantObjectLoggingTagSmoothingMode	Smoothing mode of logging tag	R
SmoothingMinTime	System.TimeSpan	Minimum time of logging tag	R
SmoothingMaxTime	System.TimeSpan	Maximum time of logging tag	R
SmoothingDeltaValue	System.Double	Delta of logging tag	R
LimitScope	PlantObjectLoggingTagLimitScope	Limit scope of logging tag	R
HighLimit	System.Object	High limit of logging tag	R
LowLimit	System. Object	Low limit of logging tag	R

Requirement

- The TIA Portal Openness is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)

Program code

Modify the following program code to access all the properties of logging tags:

```
// Instance inside plant view node
PlantViewNode mixingNode = plantView.PlantViewNodes.Create("Mixing");
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Create("IngredientMixer", "Mixer");
PlantObjectInterfaceMemberComposition interfaceTags =
mixerObject.PlantObject.PlantObjectInterfaceMembers;
PlantObjectInterfaceMember interfaceTag = interfaceTags.FirstOrDefault();
PlantObjectInterfaceMemberComposition memberTags = interfaceTag.Members;
PlantObjectInterfaceMember memberTag = memberTags.FirstOrDefault();
PlantObjectLoggingTagComposition loggingTags = memberTag.LoggingTags;
PlantObjectLoggingTag loggingTag = loggingTags.FirstOrDefault();
string name = loggingTag.Name;
object lowLimit = loggingTag.LowLimit;
```

Note

You will not be able to update any property of PlantObjectLoggingTag class, associated to a CPM object instance logging tag.

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.19 Functions for accessing the data of a PLC device

7.19.1 Determining the status of a PLC

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See AUTOHOTSPOT
- You have opened a project with your TIA Portal Openness application.
See AUTOHOTSPOT

Application

You can determine the state of a PLC or all PLCs in a project.

7.19 Functions for accessing the data of a PLC device

TIA Portal Openness distinguishes between the following states:

- Offline
- PLC is connected ("Connecting")
- Online
- PLC is disconnected ("Disconnecting")
- Incompatible
- Not reachable
- Protected

Program code

Modify the following program code to determine the state of a PLC:

```
public static OnlineState GetOnlineState(DeviceItem deviceItem)
{
    OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
    return onlineProvider.State;
}
```

Modify the following program code to determine the state of all PLCs in a project:

```
public static void DetermineOnlineStateOfAllProjectDevices(Project project)
{
    foreach (Device device in project.Devices)
    {
        foreach (DeviceItem deviceItem in device.DeviceItems)
        {
            OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
            if (onlineProvider != null)
            {
                OnlineState state = onlineProvider.State;
            }
        }
    }
}
```

7.19.2 Accessing parameters of an online connection

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See AUTOHOTSPOT
- A project is open.
See AUTOHOTSPOT

Application

You can use the TIA Portal Openness API interface to determine or set parameters for an online connection:

- Enumerate the available connection modes to a PLC
- Enumerate the available interfaces to a PLC
- Enumerate the allocated slots
- Enumerate the available addresses of the subnets and gateways
- Set the connection parameters.

Program code: Determining connection parameters

Modify the following program code to enumerate the available connection modes, PC interfaces and slots:

```
public static void EnumerateConnectionModesOfPLC(DeviceItem deviceItem)
{
    OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
    if (onlineProvider == null)
    {
        return; // Only cpu device items can provide OnlineProvider service
    }
    // Accessing connection configuration object
    ConnectionConfiguration configuration = onlineProvider.Configuration;
    // Now access connection configuration members
    foreach (ConfigurationMode mode in configuration.Modes)
    {
        Console.WriteLine("Mode name:{0}", mode.Name);
        foreach (ConfigurationPcInterface pcInterface in mode.PcInterfaces)
        {
            Console.WriteLine("PcInterface name:{0}", pcInterface.Name);
            Console.WriteLine("PcInterface number:{0}", pcInterface.Number);
            foreach (ConfigurationTargetInterface targetInterface in
pcInterface.TargetInterfaces)
            {
                Console.WriteLine("TargetInterface:{0}", targetInterface.Name);
            }
        }
    }
}
```

7.19 Functions for accessing the data of a PLC device

You can also access a connection mode and a PC interface by name:

```
public static ConfigurationTargetInterface
GetTargetInterfaceForOnlineConnection(OnlineProvider onlineProvider)
{
    ConnectionConfiguration configuration = onlineProvider.Configuration;
    ConfigurationMode mode = configuration.Modes.Find("PN/IE");
    ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("PLCSIM", 1);
    ConfigurationTargetInterface slot = pcInterface.TargetInterfaces.Find("2 X3");
    return slot;
}
```

Modify the following program code to enumerate the addresses of the subnets and gateways available on a PC interface:

```
public static void EnumeratingPCInterfaceSubnetsAndGateways(ConfigurationPcInterface
pcInterface)
{
    foreach (ConfigurationSubnet subnet in pcInterface.Subnets)
    {
        Console.WriteLine("Subnet name:{0}", subnet.Name);
        foreach (ConfigurationGateway gateway in subnet.Gateways)
        {
            //Get the name of the gateway:
            Console.WriteLine("Gateway name:{0}", gateway.Name);
            //Get the IP address of each gateway:
            foreach (ConfigurationAddress gatewayAddress in gateway.Addresses)
            {
                Console.WriteLine("Gateway Address:{0} has {1}", gatewayAddress.Name,
gatewayAddress.Address);
            }
        }
    }
}
```

You can also access subnets and gateways by their name or IP address:

```
public static void AccessSubnetAndGatewayOfPCInterface(ConfigurationPcInterface
pcInterface)
{
    ConfigurationSubnet subnet = pcInterface.Subnets.Find("PN/IE_1");
    ConfigurationAddress subnetAddress = subnet.Addresses.Find("192.168.0.1");
    ConfigurationGateway gateway = subnet.Gateways.Find("Gateway 1");
    ConfigurationAddress gatewayAddress = gateway.Addresses.Find("192.168.0.2");
}
```

Program code: Setting connection parameters

Note

All the connection parameters previously set are overwritten when you set the connection parameters. If you have already set the connection parameters directly in the TIA Portal, it is not necessary to call `ApplyConfiguration`. If there is already an online connection to a PLC while `ApplyConfiguration` is called, an exception is thrown.

Modify the following program code to set slot parameters:

```
public static void SetConnectionWithSlot(OnlineProvider onlineProvider)
{
    ConnectionConfiguration configuration = onlineProvider.Configuration;
    ConfigurationMode mode = configuration.Modes.Find(@"PN/IE");
    ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("PLCSIM", 1);
    // or network pc interface that is connected to plc
    ConfigurationTargetInterface slot = pcInterface.TargetInterfaces.Find("2 X3");
    configuration.ApplyConfiguration(slot);
    // After applying configuration, you can go online
    onlineProvider.GoOnline();
}
```

Modify the following program code to set gateway address parameters:

```
public static void SetConnectionWithGatewayAddress(OnlineProvider onlineProvider, string
subnetName, string gatewayAddressName)
{
    ConnectionConfiguration configuration = onlineProvider.Configuration;
    ConfigurationMode mode = configuration.Modes.Find(@"PN/IE");
    ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("PLCSIM", 1);
    // or network pc interface that is connected to plc
    ConfigurationSubnet subnet = pcInterface.Subnets.Find(subnetName);
    ConfigurationAddress gatewayAddress = subnet.Addresses.Find(gatewayAddressName);
    configuration.ApplyConfiguration(gatewayAddress);
    // After applying configuration, you can go online
    onlineProvider.GoOnline();
}
```

Modify the following program code to set subnet address parameters:

```
public static void SetConnectionWithSubnetAddress(OnlineProvider onlineProvider, string
subnetName)
{
    ConnectionConfiguration configuration = onlineProvider.Configuration;
    ConfigurationMode mode = configuration.Modes.Find(@"PN/IE");
    ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("PLCSIM", 1);
    // or network pc interface that is connected to plc
    ConfigurationSubnet subnet = pcInterface.Subnets.Find(subnetName);
    ConfigurationAddressComposition addresses = subnet.Addresses;
    configuration.ApplyConfiguration(addresses[0]);
    // After applying configuration, you can go online
    onlineProvider.GoOnline();
}
```

7.19.3 Accessing fingerprint for quick station compare

Requirement

- The Openness application is connected to the TIA Portal.
See AUTOHOTSPOT
- A project is open.
See AUTOHOTSPOT
- PLC is offline.

Application

You can use the TIA Portal Openness to get FingerprintData for different aspects of PLC device configuration which can be used to realize a quick station compare. It is done through FingerprintDataProvider service that can be acquired from a given TIA Portal. An instance of FingerprintDataProvider will be returned on GetService call, else the service will return null.

Program code: Retrieving FingerprintDataProvider from TIA Portal

```
TiaPortal tia = new TiaPortal(TiaPortalMode.WithUserInterface);
FingerprintDataProvider fingerprintDataProvider =
tia.GetService<FingerprintDataProvider>();
if (fingerprintDataProvider != null)
{
    ...
}
```

Parameters of FingerprintDataProvider method

To get the fingerprintData of a device, you have to call GetFingerprintData method of FingerprintDataProvider.

The following attributes are supported in FingerprintDataProvider:

Parameter name	Type	Description
configurationAddress	Siemens.Engineering.Online.Configuration	Address of device for which the finger- printData have to be fetched.
onlineConfigurationDelegate	Siemens.Engineering.Online	Delegate that will be called to check configuration before fetching the finger- printData.

Configuration Address

You should provide a ConfigurationAddress object to the GetFingerprintData. The address object will be used to establish a connection to the device for which the fingerprintData have to be fetched. The ConfigurationAddress object must be created in the ConnectionConfiguration of the FingerprintDataProvider.

For example, you can use the following code to create an address object:

```
...
ConnectionConfiguration configuration = fingerprintDataProvider.Configuration;
ConfigurationMode configurationMode = configuration.Modes.Find("PN/IE");
ConfigurationPcInterface pcInterface = configurationMode.PcInterfaces.Find("Intel(R)
Ethernet Connection I217-LM", 1);
// Create an address. This "ConfigurationAddress" is used as parameter for getting the
fingerprintData
ConfigurationAddress fingerprintAddress = pcInterface.Addresses.Create("192.68.0.1");
...
```

The Configuration contains a list of supported Modes. You have to select one of the Modes that should be used to get fingerprints. The selected ConfigurationMode contains a list of all local PCInterfaces that support the selected Mode. You have select one of the interfaces. The desired address can be created in the Address collection of the selected ConfigurationPcInterface.

Program code: Retrieving the FingerprintData

You can start retrieving the FingerprintData of the station by calling the action GetFingerprintData. The following parameters are supported:

Parameter	Description
ConfigurationAddress	The address of the device
OnlineConfigurationDelegate	The callback to handle inhibits)

```

...
OnlineConfigurationDelegate preConfigurationDelegate = PreConfigureFingerprint;
FingerprintDataResult result =
fingerprintDataProvider.GetFingerprintData(fingerprintAddress , preConfigurationDelegate);
// The fingerprints
FingerprintDataItemComposition allFingerprints = result.FingerprintDataItems;
foreach (FingerprintDataItem item in allFingerprints)
{
// do something with the fingerprint ...
}
internal void PreConfigureFingerprint(OnlineConfiguration onlineConfiguration)
{
...
}

```

Online configuration delegate

The possible online configuration types are listed below:

Configuration name	Description and properties
OnlineConfiguration	<ul style="list-style-type: none"> • Base class for other configurations • OnlineConfiguration.Message: string (read only property that contains the configuration message)

Because this is the Base class for all other configurations, this property adapter is therefore available in all configurations.

The datatype of the configuration is given below:

Configuration	DataType	Description and Action
OnlineConfiguration	OnlineReadAccessPassword	Set password via SetPassword(password:SecureString) method. Enter a password to gain read access to the module

Unhandled configuration that can prevent accessing fingerprint causes EngineeringDelegateInvocationException and aborts action. An EngineeringDelegateInvocationException will be thrown in case of an unhandled exception within the Delegate.

PreConfigureFingerprint implementation example

```
private static void PreConfigureFingerprint(OnlineConfiguration onlineConfiguration)
{
    OnlineReadAccessPassword readAccess = onlineConfiguration as OnlineReadAccessPassword;
    if (readAccess != null)
    {
        string passWD = "passWD";
        var password = new SecureString();
        foreach (var c in passWD) password.AppendChar(c);
        moduleReadAccessPassword.SetPassword(password);
        return;
    }
    throw new NotSupportedException(); // Exception thrown in the delegate will cancel get
    fingerprints
}
```

FingerprintDataResult

The FingerprintDataResult returned by the GetFingerprintData action provides a composition of all the fingerprints defined for the device.

For example, you can use the below code to examine fingerprint items:

```
{
...
FingerprintDataResult result = dataProvider.GetFingerprintData(fingerprintAddress,
preConfigurationDelegate);
FingerprintDataItemComposition allFingerprints = result.FingerprintDataItems;
foreach (FingerprintDataItem item in allFingerprints)
{
    string fingerprintDataIdentifier = item.FingerprintDataIdentifier;
    string fingerprintDataValue = plcItem.FingerprintDataValue;
    // Do something, e.g. store Identifier and Value in a file
}
}
```

Error Handling within the Openness fingerprints

All errors within the Openness fingerprints will be mapped as recoverable user exceptions EngineeringTargetInvocationException. So any error in an Openness action will not lead to a TIA-Portal crash.

Precondition is, that the client handle the exception. For example this could happen like shown in following code

```
TiaPortal currentTIA = ...;
FingerprintDataProvider dataProviderFingerprints =
currentTIA.GetService<FingerprintDataProvider>();
OnlineConfigurationDelegate configurationDelegate = OnlineConfigurationFingerprintData;
try
{
// The used addr refers a PLC300.
FingerprintDataResult dataResult = dataProviderFingerprints.GetFingerprintData(addrObj,
configurationDelegate);
// Try to get a list of all FingerprintDataItems. But a PLC300 supports no fingerprints.
FingerprintDataItemComposition dataItems = dataResult.FingerprintDataItems;
foreach (FingerprintDataItem searchItem in dataItems)
{
string valueToCompareIdent = searchItem.FingerprintDataIdentifier;
string valueToCompareValue = searchItem.FingerprintDataValue;
DoAnything(valueToCompareIdent, valueToCompareValue);
}
}
catch (EngineeringTargetInvocationException targetException)
{
// In case of the not supported PLC this exception will be caught.
Console.WriteLine(targetException.Message);
}
```

Helper classes

For this example two classes are needed. You can use the below program code to encapsulate fingerprint data pairs.

```
/// <summary>
/// Helperclass to handle a "Dictionary" as serializeable class object
/// </summary>
[Serializable]
public class FingerprintDataPairs : Dictionary<string, string>, ISerializable
{
public FingerprintDataPairs(){}
public FingerprintDataPairs(SerializationInfo si, StreamingContext context)
{
KeyValuePair<string, string>[] dataPair = si.GetValue("KeyValuePair",
typeof(KeyValuePair<string, string>[])) as KeyValuePair<string, string>[];
if (dataPair != null)
foreach (KeyValuePair<string, string> pair in dataPair)
this.Add(pair.Key, pair.Value);
}
}
```

You can use the below example to handle the fingerprint data and compare:

```
/// <summary>
/// Helperclass for handle the fingerprint data and compare two datasets.
/// </summary> public class FingerprintDataCompare
{
internal FingerprintDataPairs m_FingerprintDataPairs;
internal FingerprintDataPairs FingerprintDataPairSet
{
get
{
if (m_FingerprintDataPairs == null) m_FingerprintDataPairs = new FingerprintDataPairs();
return m_FingerprintDataPairs;
}
set
{
m_FingerprintDataPairs = value;
}
}
internal bool CompareDataSets(FingerprintDataPairs dataSet1, FingerprintDataPairs dataSet2)
{
foreach (string key in dataSet1.Keys)
{
if (dataSet2[key] != dataSet1[key])
return false;
}
return true;
}
internal void SaveFingerprintDataItems(FingerprintDataPairs fingerprintDataPairSet, string filename)
{
using (FileStream fs = new FileStream(filename, FileMode.Create))
{
BinaryFormatter formatter = new BinaryFormatter();
try
{
formatter.Serialize(fs, fingerprintDataPairSet);
}
catch (SerializationException e)
{
Console.WriteLine("Failed to write fingerprints. See: " + e.Message);
}
finally
{
fs.Close();
}
}
}
internal FingerprintDataPairs LoadFingerprintDataItems(string filename)
{
FingerprintDataPairs dataPair = null;
using(FileStream fs = new FileStream(filename, FileMode.Open))
{
BinaryFormatter formatter = new BinaryFormatter();
try
{
dataPair = (FingerprintDataPairs)formatter.Deserialize(fs);
}
```

```
        }
    catch (System.Runtime.Serialization.SerializationException e)
    {
        Console.WriteLine("Failed to read fingerprints. See: " + e.Message);
    }
    finally
    {
        fs.Close();
    }
}
return dataPair;
}
```

Compare Fingerprints

To compare, you can use the `FingerprintDataResult`

```
...
// See chapter "FingerprintDataResult"
FingerprintDataResult fingerprintDataResult = dataProvider.GetFingerprintData(address,
preConfigurationDelegate);
...
```

To save the Fingerprints. you can use the Helper Classes:

```
string pathDat = "FingerprintDataOverview.dat";
// Create a fingerprint comparer and fill the dataitems with the result of
// "GetFingerprintData".
FingerprintDataCompare fingerprintDataSet = new FingerprintDataCompare();
foreach (FingerprintDataItem item in fingerprintDataResult.FingerprintDataItems)
    fingerprintDataSet.FingerprintDataPairSet.Add(item.FingerprintDataIdentifier,
item.FingerprintDataValue);
fingerprintDataSet.SaveFingerprintDataItems(fingerprintDataSet.FingerprintDataPairSet,
pathDat);
```

7.19 Functions for accessing the data of a PLC device

With saved FingerprintDataItems and a current FingerprintDataResult a Compare of Fingerprints can be realized

```
...
string pathDat = "FingerprintDataOverview.dat";
FingerprintDataCompare fingerprintDataSet = new FingerprintDataCompare();
foreach (FingerprintDataItem item in fingerprintDataResult.FingerprintDataItems)
    fingerprintDataSet.FingerprintDataPairSet.Add(item.FingerprintDataIdentifier,
        item.FingerprintDataValue);
FingerprintDataPairs loadDataSet = new FingerprintDataPairs();
loadDataSet = fingerprintDataSet.LoadFingerprintDataItems(pathDat);
if (!fingerprintDataSet.CompareDataSets(loadDataSet,
    fingerprintDataSet.FingerprintDataPairSet))
{
// Here is place to implement user actions.
}
```

7.19.4 Accessing CM DP as DP slave and transfer area

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal
- A project is open
See Opening a project

Application

You can use the TIA Portal Openness to configure an ET200SP PLC as DP slave if a CM DP has been plugged. It is also possible in TIA Portal Openness to create, configure and delete transfer areas at the DP interface. The handling will be similar to the handling of transfer areas at PN interface described in Openness transfer areas for PnPn coupler

The following dynamic attributes are supported at the device item DP interface.

Attribute name	Data type	Access	Writable
DpUseForTestCommissioning- Routing	Boolean	only available if config- ured as DP slave	r/w
DpWatchdog	Boolean	only available if config- ured as DP slave and assigned to DP Master	r/w

Program code: Creating transfer areas

Modify the following program code to create a transfer area at the DP interface:

```
NetworkInterface cpuItf = CpuInterface.GetService<NetworkInterface>();
// Create TransferAreas
TransferAreaComposition transferAreas = cpuItf.TransferAreas;
// Simple TranferArea
TransferArea transferAreaExample = transferAreas.Create("Example TA MS",
TransferAreaType.MS);
```

Parameter of transfer areas

The following parameters are supported in transfer areas of DP slave configuration:

Parameter name	Data type	Access
Name	string	read/write
Direction	enum	read/write
Comment	string	read/write
LocalToPartnerLength	Int32	read/write
PartnerToLocalLength	Int32	read/write
LocalAddresses	AddressComposition object	read
PartnerAddresses	AddressComposition object	read
PositionNumber	Int32	read
Type	enum	read

Program code: Deleting transfer area

Modify the following program code to delete transfer area for DP slave configuration:

```
TransferArea transferAreaExample = transferAreas.Create("Example TA MS",
TransferAreaType.MS);
transferAreaExample.Delete();
```

7.19.5 Setting PLC online of R/H system

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal
- A project is open.
See Opening a project

Application

You can use the RHOnlineProvider service to set online either to primary PLC or backup PLCs of R/H system.

Program code: Accessing RHOnlineProvider service from a device

Modify the following code to access RHOnlineProvider:

```
Device device = project.Devices.Find("S7-1500R/H-System_1");
RHOnlineProvider rhOnlineProvider = device.GetService<RHOnlineProvider>();
```

Program code: Setting connection parameters

You can use ConnectionConfiguration object to set a connection to the device. It can be accessed from Configuration property of the RHOnlineProvider. For more information about how to set connection, Refer Accessing parameters of an online connection

Modify the following program code to set a connection mode and access a PC interface by name:

```
ConnectionConfiguration connectionConfiguration = rhOnlineProvider.Configuration;
ConfigurationMode mode = connectionConfiguration.Modes.Find("PN/IE");
ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("Broadcom NetXtreme Gigabit Ethernet", 1);
ConfigurationTargetInterface targetConfiguration = pcInterface.TargetInterfaces.Find("1 X1");
bool success = connectionConfiguration.ApplyConfiguration(targetConfiguration);
```

Note

R/H system consists of two PLCs, a single connection configuration is provided to you.

Program code: Setting online R/H system

You can set online either to primary or backup PLC. The user attempts to set online to both targets simultaneously will encounter EngineeringTargetExceptionException from system.

Modify the following program code to set online to the primary PLC:

```
OnlineState onlineState = rhOnlineProvider.GoOnlineToPrimary();
```

Modify the following program code to set online to the backup PLC:

```
OnlineState onlineState = rhOnlineProvider.GoOnlineToBackup();
```

Note

You are allowed to reuse previously stored password when setting a PLC Online of R/H system.

Program code: Determining online status of R/H system

You can use PrimaryState and BackupState properties of RHOnlineProvider to determine the online connection status of primary PLC and backup PLC individually . Both properties return enum OnlineState. For more information on identify the online state of PLC, Refer Determining the status of a PLC

Modify the following program code to determine the state of primary PLC and backup PLC:

```
RHOnlineProvider rhOnlineProvider = ...;
OnlineState primaryState = rhOnlineProvider.PrimaryState;
OnlineState backupState = rhOnlineProvider.BackupState;
```

Program code: Setting offline R/H system

Modify the following program code to set a currently online R/H system to an offline state by invoking RHOnlineProvider.GoOffline method:

```
rhOnlineProvider.GoOffline();
```

7.19.6 Accessing software container from primary PLC of R/H system

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal
- A project is open.
See Opening a project

Application

You can use primary PLC device of an R/H system to access software container, for example the R/H system will provide software container for a primary PLC device item representing PLC_1. Otherwise, it will provide null if you try to access a software container for backup PLC device representing PLC_2.

The specific of SoftwareContainer and its software property are described in Access software target.

Program code: Accessing software container

Modify the following program code to access software container from primary device of an R/H system:

```
foreach (DeviceItem deviceItem in rhDevice.DeviceItems)
{
    if (deviceItem.Name == "PLC_1")
    {
        SoftwareContainer softwareContainer = deviceItem.GetService<SoftwareContainer>();
        ... //Work with softwareContainer
    }
}
```

7.19.7 Downloading PLCs of R/H System

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal
- A project is open.
See Opening a project

Application

You can use the TIA Portal Openness application to download the primary and backup PLCs of R/H system. You are able to download both hardware and software components of the system. (Refer Downloading hardware and software components to PLC device)

Program code: Retrieving RHDownloadProvider

You can download to R/H system through RHDownloadProvider service from a device.

Modify the following program code to retrieve RHDownloadProvider:

```
...
Device device = project.Devices.Find("S7-1500R/H-System_1");
RHDownloadProvider rhDownloadProvider = device.GetService<RHDownloadProvider>();
...
```

Note

The DownloadProvider service will not be accessed for CPUs that are part of R/H system.

Program code: Retrieving IConfiguration

RHDownloadProvider provides ConnectionConfiguration object through Configuration property which will be used to configuring the connection to the device.

Modify the following program code to retrieve IConfiguration object from ConnectionConfiguration on RHDownloadProvider:

```
...
RHDownloadProvider rhDownloadProvider = device.GetService<RHDownloadProvider>();
ConnectionConfiguration connectionConfiguration = rhDownloadProvider.Configuration;
ConfigurationMode mode = connectionConfiguration.Modes.Find("PN/IE");
ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("Broadcom NetXtreme Gigabit
Ethernet", 1);
IConfiguration targetConfiguration = pcInterface.TargetInterfaces.Find("1 X1");
...

```

Note

R/H systems consist of two PLCs, only one connection configuration object is provided that can be used for both primary and backup downloads.

Program code: Downloading primary CPU and backup CPU

Modify the following program code to download to the primary CPU by invoking RHDownloadProvider.DownloadToPrimary:

```
DownloadResult DownloadToPrimary(configuration, preDownloadConfigurationDelegate,
postDownloadConfigurationDelegate, downloadOptions);
```

Modify the following program code to download to the backup CPU by invoking RHDownloadProvider.DownloadToBackup:

```
DownloadResult DownloadToBackup(configuration, preDownloadConfigurationDelegate,
postDownloadConfigurationDelegate, downloadOptions);
```

Parameters of RHDownloadProvider method

Both RHDownloadProvider.DownloadToPrimary and RHDownloadProvider.DownloadToBackup accept the same parameters and also return a DownloadResult. For more information about the details of IConfiguration, DownloadConfigurationDelegate, DownloadOptions and DownloadResult, Refer Downloading hardware and software components to PLC device

Parameter name	Type	Description
configuration	Siemens.Engineering.Connection.IConfiguration	Connection configuration to a device.
preDownloadConfigurationDelegate	Siemens.Engineering.Download.DownloadConfigurationDelegate	Delegate that will be called to check configuration before download

7.19 Functions for accessing the data of a PLC device

Parameter name	Type	Description
postDownloadConfigurationDelegate	Siemens.Engineering.Download.DownloadConfigurationDelegate	Delegate that will be called to check configuration after download
downloadOptions	Siemens.Engineering.Download.DownloadOptions	Download options

Depending upon the state of R/H system, you might request to stop the system for the download via DownloadConfigurations. Therefore, in addition to the configuration described in Downloading hardware and software components to PLC device, the following data type are added to support RHDownload.

Configuration	Data Type	Action	Description
DownloadSelection-Configuration	StopHSystem	Set CurrentSelection:StopH-SystemSelections. Available enum values: <ul style="list-style-type: none"> • NoAction (No Action) • StopHSystem (Stop R/H-System) 	The modules are stopped for downloading to device
	StopHSystemOrModule	Set CurrentSelection:StopH-SystemOrModuleSelections. Available enum values: <ul style="list-style-type: none"> • NoAction (No action) • StopHSystem (Stop R/H-System) • StopModule (Stop module) 	The modules are stopped for downloading to device
	StartBackupModules	Set CurrentSelection:Start-BackupModulesSelections. Available enum values: <ul style="list-style-type: none"> • NoAction (No action) • SwitchToPrimaryCpu (Change to Primary)) • StartModule (Start module) 	Start modules after downloading to device
	SwitchBackupToPrimary	Set CurrentSelection:Switch-BackupToPrimarySelections. Available enum values: <ul style="list-style-type: none"> • NoAction (No action) • SwitchToPrimaryCpu (Change to Primary) 	Start modules after downloading to device.

Program code: Handling download configuration callbacks

Modify the following program code to DownloadToPrimary and DownloadToBackup invocations while handling configurations in the callbacks:

Download invocation example

```

static void Main(string[] args)
{
    ...
    Project project = tiaPortal.Projects[0];
    Device device = project.Devices.Find("S7-1500R/H-System_1");
    RHDownloadProvider rhDownloadProvider = device.GetService<RHDownloadProvider>();
    ConnectionConfiguration connectionConfiguration = rhDownloadProvider.Configuration;
    ConfigurationMode mode = connectionConfiguration.Modes.Find("PN/IE");
    ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("Broadcom NetXtreme Gigabit
Ethernet", 1);
    IConfiguration targetConfiguration = pcInterface.TargetInterfaces.Find("1 X1");

    // Download to primary
    DownloadResult primaryDownloadResult =
    rhDownloadProvider.DownloadToPrimary(targetConfiguration,
    PreConfigureDownloadCallback,
    PostConfigureDownloadCallback,
    DownloadOptions.Hardware | DownloadOptions.Software);
    WriteDownloadResults(primaryDownloadResult);
    // Download to backup
    DownloadResult backupDownloadResult =
    rhDownloadProvider.DownloadToBackup(targetConfiguration,
    PreConfigureDownloadCallback,
    PostConfigureDownloadCallback,
    DownloadOptions.Hardware | DownloadOptions.Software);
    WriteDownloadResults(backupDownloadResult);
    ...
}
private static void PreConfigureDownloadCallback(DownloadConfiguration
downloadConfiguration)
{
    StopHSystem stopHSystem = downloadConfiguration as StopHSystem;
    if (stopHSystem != null)
    {
        stopHSystem.CurrentSelection = StopHSystemSelections.StopHSystem;
    }
    OverwriteTargetLanguages overwriteTargetLanguages = downloadConfiguration as
OverwriteTargetLanguages;
    if (overwriteTargetLanguages != null)
    {
        overwriteTargetLanguages.Checked = true;
    }
    AlarmTextLibrariesDownload alarmTextLibraries = downloadConfiguration as
AlarmTextLibrariesDownload;
    if (alarmTextLibraries != null)
    {
        alarmTextLibraries.CurrentSelection =
AlarmTextLibrariesDownloadSelections.ConsistentDownload;
        return;
    }
    CheckBeforeDownload checkBeforeDownload = downloadConfiguration as CheckBeforeDownload;
    if (checkBeforeDownload != null)
    {
        checkBeforeDownload.Checked = true;
        return;
    }
}

```

Download invocation example

```

        }
        ConsistentBlocksDownload consistentBlocksDownload = downloadConfiguration as
ConsistentBlocksDownload;
        if (consistentBlocksDownload != null)
        {
            consistentBlocksDownload.CurrentSelection =
ConsistentBlocksDownloadSelections.ConsistentDownload;
            return;
        }
        OverwriteSystemData overwriteSystemData = downloadConfiguration as OverwriteSystemData;
        if (overwriteSystemData != null)
        {
            overwriteSystemData.CurrentSelection = OverwriteSystemDataSelections.Overwrite;
            return;
        }
    }
    private static void PostConfigureDownloadCallback(DownloadConfiguration
downloadConfiguration)
{
    StartModules startModules = downloadConfiguration as StartModules;
    if (startModules != null)
    {
        startModules.CurrentSelection = StartModulesSelections.StartModule;
        return;
    }
}
private static void WriteDownloadResults(DownloadResult result)
{
    Console.WriteLine("State:" + result.State);
    Console.WriteLine("Warning Count:" + result.WarningCount);
    Console.WriteLine("Error Count:" + result.ErrorCount);
    RecursivelyWriteMessages(result.Messages);
}
private static void RecursivelyWriteMessages(DownloadResultMessageComposition messages,
string indent = "")
{
    indent += "\t";
    foreach (DownloadResultMessage message in messages)
    {
        Console.WriteLine(indent + "DateTime: " + message.DateTime);
        Console.WriteLine(indent + "State: " + message.State);
        Console.WriteLine(indent + "Message: " + message.Message);
        RecursivelyWriteMessages(message.Messages, indent);
    }
}

```

7.19.8 Functions for downloading data to PLC device

7.19.8.1 Downloading PC System

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- You have opened a project with your TIA Portal Openness application.
See Opening a project (Page 106)

Application

You can download PC-Station Plus and the SW-CPU independently. In order to download the PC-Station Plus configuration, you must retrieve stationmanager device item of rack of device and get the download provider as a service. SW-CPU itself must be retrieved as a device item for the same operation.

Program code: Download software and hardware components

You are able to download software and hardware components to a device via DownloadProvider service that can be acquired from a given DeviceItem. If a DeviceItem represents a downloadable target an instance of DownloadProvider will be returned on GetService call. Otherwise it will be null.

```
DeviceItem stationManager = dev.DeviceItems.First(p => p.PositionNumber == 0).DeviceItems.First(a => a.PositionNumber == 125);
DownloadProvider downloadProviderStationManager =
stationManager.GetService<DownloadProvider>();
if (downloadProviderStationManager == null)
{
    //no download is possible for PC-Station
}
DeviceItem swCpu = dev.DeviceItems.First(p => p.Name == "Software PLC_1");
DownloadProvider downloadProviderSwCpu = swCpu.GetService<DownloadProvider>();
```

Modify the following program code to configure Network parameters:

```
ConnectionConfiguration connConfig = downloadProviderStationManager.Configuration;
ConfigurationMode configurationMode = connConfig.Modes.Find("PN/IE");
ConfigurationPcInterface pcInterface = configurationMode.PcInterfaces.Find("ASIX AX88179
USB 3.0 to Gigabit Ethernet Adapter", 1);
ConfigurationTargetInterface targetInterface = pcInterface.TargetInterfaces.Find("2 X2");
IConfiguration targetConfiguration = pcInterface.TargetInterfaces[0];
bool isConfigured = connConfig.ApplyConfiguration(targetInterface);
if (isConfigured)
...
...
```

It must be noted that, in case the target system reboots after downloading PC-Station Plus configuration, the default behavior in openness is **waiting**. This means that, the openness code flow is stopped until the first stage download is successful (and target system is rebooted) or timed out or failed.

In case you do not want to wait for the reboot, you can use a post download option which is called `WaitOnReboot`

```
//Post Download Configuration Delegate
DownloadConfigurationDelegate postDownloadForPcStation = downloadConfiguration =>
{
    WaitOnReboot waitOnReboot = downloadConfiguration as WaitOnReboot;
    if (waitOnReboot != null)
    {
        //In case user does not want to wait...
        waitOnReboot.CurrentSelection = WaitOnRebootSelections.NoAction;
        //In case user wants to wait... This is the default option anyway...
        //waitOnReboot.CurrentSelection = WaitOnRebootSelections.Wait;
        return;
    }
};
```

You must continue downloading SW-CPU separately, but it must be made sure that the download of PC-Station was executed before.

```
DownloadResult downloadResult = null;
try
{
    //WE FIRST DOWNLOAD PC-STATION
    downloadResult = downloadProviderStationManager.Download(targetConfiguration,
        preDownloadForPcStation, postDownloadForPcStation, DownloadOptions.Hardware);
    if (DownloadResultState.Error != downloadResult.State)
    {
        Console.WriteLine("The download is successful for pc-station");
    }
}
catch (EngineeringTargetInvocationException e)
{
    Console.WriteLine("Exception Thrown, Message: " + e.Message.ToString());
}
downloadResult = null;
try
{
    downloadResult = downloadProviderSwCpu.Download(targetConfiguration, preDownloadForSwCpu,
        postDownloadForWinac, DownloadOptions.Hardware | DownloadOptions.Software);
    if (DownloadResultState.Error != downloadResult.State)
    {
        Console.WriteLine("The download is successful for SW-CPU");
    }
}
catch (EngineeringTargetInvocationException e)
{
    Console.WriteLine("Exception Thrown, Message: " + e.Message.ToString());
}
```

7.19 Functions for accessing the data of a PLC device

Downloading SW-CPU is similar to downloading a regular plc. For downloading SW-CPU, please see Hotspot-Text (Page 450).

Note

Download of F-Activated PLCs is not currently supported for SW-CPU as well; as it is in HW-CPU.

A download method also triggers compilation. It is also possible to compile a device item stand alone, with the following code snippet:

```
ICompilable compileServiceSwCpu = swCpu.GetService<ICompilable>();  
ICompilable compileServiceStationManager = stationManager.GetService<ICompilable>();  
CompilerResult compileResultStationManager = compileServiceStationManager.Compile();  
CompilerResult compileResultSwCpu = compileServiceStationManager.Compile();  
bool compileCheck = !compileResultStationManager.State.Equals(CompilerResultState.Error);  
if (compileCheck != true)  
{  
    ...  
}
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

[Downloading hardware and software components to PLC device \(Page 450\)](#)

7.19.8.2 Downloading hardware and software components to PLC device

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- You have opened a project with your TIA Portal Openness application.
See [Opening a project \(Page 106\)](#)

Application

Openness user is able to download software and hardware components to PLC device through DownloadProvider (accessed from the DeviceItem). If a DeviceItem represents a downloadable target, an instance of DownloadProvider will be returned on GetService call, else the service will return null.

Program code: Retrieving DownloadProvider service from a device item

```
DeviceItem deviceItem = ...;
DownloadProvider downloadProvider = deviceItem.GetService<DownloadProvider>();
if (downloadProvider != null)
{
    ...
}
```

Parameters of download method

In order to download to a PLC device, the user calls Download method of DownloadProvider. The Download method has four parameters which are IConfiguration object, two delegates and DownloadOptions (Hardware, Software or Hardware and Software).

Parameter name	Type	Description
configuration	Siemens.Engineering.Connection.IConfiguration	Connection configuration to a device.
preDownloadConfigurationDelegate	Siemens.Engineering.Download.DownloadConfigurationDelegate	Delegate to be called for checking configuration before download operation.
postDownloadConfigurationDelegate	Siemens.Engineering.Download.DownloadConfigurationDelegate	Delegate to be called for checking configuration after download operation.
downloadOptions	Siemens.Engineering.Download.DownloadOptions	Download options.

- Openness download is supported only if the configurations are handled properly by the user. If the configuration is invalid, then EngineeringTargetInvocationException is thrown and download process is aborted. The F-activated PLCs are not supported for download operation.
- Since compilation is a part of download, it is recommended to compile before the download operation to analyze the compile results.
- Openness supports only full download option.

Parameter 1: IConfiguration

The user should provide IConfiguration object as first parameter for the Download method. It is used to establish a connection to the given PLC device. IConfiguration interface is implemented by ConfigurationAddress and ConfigurationTargetInterface. Both the objects can be accessed through ConnectionConfiguration instance. ConnectionConfiguration instance can be acquired from DownloadProvider.Connection: ConnectionConfiguration or optionally from OnlineProvider.Connection: ConnectionConfiguration properties.

Configuration of ConnectionConfiguration object is described in AUTOHOTSPOT section.

```
...
DownloadProvider downloadProvider = null;
ConnectionConfiguration configuration = downloadProvider.Configuration;
ConfigurationMode configurationMode = configuration.Modes.Find("PN/IE");
ConfigurationPcInterface pcInterface = configurationMode.PcInterfaces.Find("Intel(R)
Ethernet Connection I217-LM", 1);
IConfiguration targetConfiguration = pcInterface.TargetInterfaces[0];
...
...
```

Parameter 2 and 3: DownloadConfigurationDelegate

Openness user need to provide two implementations of void DownloadConfigurationDelegate(DownloadConfiguration downloadConfiguration). First delegate will be called for pre-download configurations, and the second will be called after download is completed. The delegates will be called for each configuration that requires an action from the user. For more information about callback handling, Refer Supporting callbacks (Page 461). Certain configurations will only contain an information, therefore the user action will not be required.

The possible download configuration types are listed below.

Configuration name	Description and properties
DownloadConfiguration	<ul style="list-style-type: none"> • Base class for all the configurations. • Contains single property DownloadConfiguration.Message : string (read only property contains the configuration message)
DownloadSelectionConfiguration	<ul style="list-style-type: none"> • Base class for all configuration that can be selected. • Does not contain additional properties. A selection must be provided in all child classes derived from it.
DownloadCheckConfiguration	<ul style="list-style-type: none"> • Base class for all configuration that can be checked and unchecked. • Contains single property DownloadCheckConfiguration.Checked: bool<string> (read/write property identifies whether the configuration is checked or unchecked)
DownloadPasswordConfiguration	<ul style="list-style-type: none"> • Base class for all configuration that required a password for download. • Contains a single method to set password. DownloadPasswordConfiguration.SetPassword (password: SecureString) : void

The datatype of the configurations are given below.

7.19 Functions for accessing the data of a PLC device

Configuration	DataType	Description and Action
DownloadSelection-Configuration	StartModules	<p>Set CurrentSelection:StopModulesSelections. Available enum values are</p> <ul style="list-style-type: none"> NoAction (No action) StartModule (Stop module) <p>These modules are stopped for downloading to a device.</p>
	StopModules	<p>Set CurrentSelection:StartModulesSelections. Available enum values:</p> <ul style="list-style-type: none"> NoAction (No action) StopAll (Stop all) <p>These modules are started after the download operation.</p>
	AllBlocksDownload	<p>Set CurrentSelection:AllBlocksDownloadSelections. Available enum value is</p> <ul style="list-style-type: none"> DownloadAllBlocks (Download all blocks to the device) <p>Downloads software to device</p>
	OverwriteSystemData	<p>Set CurrentSelection:OverwriteSystemDataSelections. Available enum values are</p> <ul style="list-style-type: none"> NoAction (No action) Overwrite (Download to device) <p>Deletes and replaces the existing system data in target location.</p>
	ConsistentBlocksDownload	<p>Set CurrentSelection:ConsistentBlocksDownloadSelections. Available enum value is</p> <ul style="list-style-type: none"> ConsistentDownload (Consistent download) <p>Downloads the software to device.</p>
	AlarmTextLibrariesDownload	<p>Set CurrentSelection:AlarmTextLibrariesDownloadSelections. Available enum values are</p> <ul style="list-style-type: none"> ConsistentDownload (Consistent download) NoAction (No action) <p>Downloads all alarm texts and text list texts.</p>
	ProtectionLevelChanged	<p>Set CurrentSelection:ProtectionLevelChangedSelections. Available enum values are</p> <ul style="list-style-type: none"> NoChange (No change) ContinueDownloading (Continue downloading to the device) <p>CPU protection is changed to the next lower level.</p>
	ActiveTestCanBeAborted	<p>Set CurrentSelection:ActiveTestCanBeAbortedSelections. Available enum values are</p> <ul style="list-style-type: none"> NoAction (No action) AcceptAll (Accept all) <p>Active test and commissioning functions are canceled during the loading operation of the device.</p>
	ResetModule	<p>Set CurrentSelection:ResetModuleSelections Available enum values are</p>

Configuration	DataType	Description and Action
		NoAction (No action) DeleteAll (Delete all) It resets the module.
	LoadIdentificationData	Set CurrentSelection:LoadIdentificationDataSelections. Available enum values are LoadNothing (Load nothing) LoadData (Load data) LoadSelected (Load selected) Load identification data to the PROFINET IO devices and their modules.
	DifferentTargetConfiguration	Set CurrentSelection:DifferentTargetConfigurationSelections. Available enum values are NoAction (No action) AcceptAll (Accept all) Gives the difference between configured and target modules (online)
	InitializeMemory	Set CurrentSelection:InitializeMemorySelections. Available enum values: NoAction (No action) AcceptAll (Accept all) This datatype is used to initialize memory.
	ExpandDownload	Set CurrentSelection: ExpandDownloadSelections. Available enum values: NoAction (No action) Download (Download) } } The download must be expanded beyond your selection.
DownloadCheckConfiguration	CheckBeforeDownload	Set IsChecked:bool property. Checks before downloading to the device.
	UpgradeTargetDevice	Set IsChecked:bool property. Checks the different project versions in the configured device and target device (online).
	OverWriteHMIData	Set IsChecked:bool property. Overwrites the objects online.
	FitHMIComponents	Set IsChecked:bool property. Components with a different version are installed on the target device.
	TurnOffSequence	Set IsChecked:bool property Turns off the sequence before loading.
	OverwriteTargetLanguages	Set IsChecked:bool property. To distinguish the settings between the project and PLC programming
	DowngradeTargetDevice	Set IsChecked:bool property. To mention the different data formats in online and offline projects.

7.19 Functions for accessing the data of a PLC device

Configuration	DataType	Description and Action
DownloadPassword-Configuration	ModuleReadAccessPassword	<p>Set password via <code>SetPassword(password:SecureString)</code> method.</p> <p>Enter a password to gain read access to the module.</p>
	ModuleWriteAccessPassword	<p>Set password via <code>SetPassword(password:SecureString)</code> method.</p> <p>Enter a password to gain write access to the module.</p>
	BlockBindingPassword	<p>Set password via <code>SetPassword(password:SecureString)</code> method.</p> <p>Block binding password configuration method.</p>

NOTICE

Please note that download configurations are similar to configurations encountered in Load preview and Load results dialogs while working with GUI of TIA Portal.

**WARNING**

The API user is responsible for ensuring the security measures of handling passwords through code.

Unhandled configuration that can prevent the download causes an `EngineeringTargetInvocationException` and aborts download. An

`EngineeringDelegateInvocationException` will be thrown in case of an unhandled exception within the Delegate.

PreDownloadDelegate implementation example:

```
private static void PreConfigureDownload(DownloadConfiguration downloadConfiguration)
{
    StopModules stopModules = downloadConfiguration as StopModules;
    if (stopModules != null)
    {
        stopModules.CurrentSelection = StopModulesSelections.StopAll; // This selection
will set PLC into "Stop" mode
        return;
    }
    AlarmTextLibrariesDownload alarmTextLibraries = downloadConfiguration as
AlarmTextLibrariesDownload;
    if (alarmTextLibraries != null)
    {
        alarmTextLibraries.CurrentSelection =
AlarmTextLibrariesDownloadSelections.ConsistentDownload;
        return;
    }
    BlockBindingPassword blockBindingPassword = downloadConfiguration as
BlockBindingPassword;
    if(blockBindingPassword != null)
    {
        SecureString password = ...; // Get Binding password from a secure location
        blockBindingPassword.SetPassword(password);
        return;
    }
    CheckBeforeDownload checkBeforeDownload = downloadConfiguration as
CheckBeforeDownload;
    if (checkBeforeDownload != null)
    {
        checkBeforeDownload.Checked = true;
        return;
    }
    ConsistentBlocksDownload consistentBlocksDownload = downloadConfiguration as
ConsistentBlocksDownload;
    if (consistentBlocksDownload != null)
    {
        consistentBlocksDownload.CurrentSelection =
ConsistentBlocksDownloadSelections.ConsistentDownload;
        return;
    }
    ModuleWriteAccessPassword moduleWriteAccessPassword = downloadConfiguration as
ModuleWriteAccessPassword;
    if (moduleWriteAccessPassword != null)
    {
        SecureString password = ...; // Get PLC protection level password from a secure
location
        moduleWriteAccessPassword.SetPassword(password);
        return;
    }
    throw new NotSupportedException(); // Exception thrown in the delegate will cancel
download
}
```

PostDownloadDelegate implementation example:

```
private static void PostConfigureDownload(DownloadConfiguration downloadConfiguration)
{
    StartModules startModules = downloadConfiguration as StartModules;
    if (startModules != null)
    {
        startModules.CurrentSelection = StartModulesSelections.StartModule; // Sets PLC in "Run" mode
    }
}
```

Parameter 4: DownloadOptions

The user must specify the download options through DownloadOptions flagged enum. This parameter will determine the type of download to be performed such as Hardware, Software or Hardware and Software.

```
[Flags]
public enum DownloadOptions
{
    None = 0,      // Download nothing
    Hardware,     // Download hardware only
    Software      // Download software only
}
```

If the user wants to download both Software and Hardware to the device, then pass DownloadOptions.Hardware | DownloadOptions.Software as the 4th parameter of the Download method.

DownloadResult

The DownloadResult returned by the Download action provides feedback on the state of the objects that were downloaded.

Download invocation example

```
[STAThread]
static void Main()
{
    ...
    DownloadProvider downloadProvider = ...;
    IConfiguration targetConfiguration = ...;
    DownloadConfigurationDelegate preDownloadDelegate = PreConfigureDownload;
    DownloadConfigurationDelegate postDownloadDelegate = PostConfigureDownload;
    DownloadResult result = downloadProvider.Download(targetConfiguration,
preDownloadDelegate, postDownloadDelegate, DownloadOptions.Hardware |
DownloadOptions.Software);
    if (result.State == DownloadResultState.Error)
    {
        // Handle error state
    }
    WriteDownloadResults(result);
    ...
}
private static void PreConfigureDownload(DownloadConfiguration downloadConfiguration)
{
    ...
}
private static void PostConfigureDownload(DownloadConfiguration downloadConfiguration)
{
    ...
}
private void WriteDownloadResults(DownloadResult result)
{
    Console.WriteLine("State:" + result.State);
    Console.WriteLine("Warning Count:" + result.WarningCount);
    Console.WriteLine("Error Count:" + result.ErrorCount);
    RecursivelyWriteMessages(result.Messages);
}
private void RecursivelyWriteMessages(DownloadResultMessageComposition messages, string
indent = "")
{
    indent += "\t";
    foreach (DownloadResultMessage message in messages)
    {
        Console.WriteLine(indent + "DateTime: " + message.DateTime);
        Console.WriteLine(indent + "State: " + message.State);
        Console.WriteLine(indent + "Message: " + message.Message);
        RecursivelyWriteMessages(message.Messages, indent);
    }
}
```

Modify the following code to download PLC by calling applying configuration:

```
private static void DownloadNCU(Device ncu, ConfigurationTargetInterface configurationTargetInterface)
{
    DownloadConfigurationDelegate preDownloadDelegate = PreConfigureDownload;
    DownloadConfigurationDelegate postDownloadDelegate = PostConfigureDownload;
    DownloadProvider downloadProvider = null;
    foreach (var item in ncu.DeviceItems[0].DeviceItems)
    {
        downloadProvider = item.GetService<DownloadProvider>();
        if (downloadProvider != null)
        {
            break;
        }
    }
    downloadProvider.Configuration.ApplyConfiguration(configurationTargetInterface);
    IConfiguration targetConfiguration = configurationTargetInterface;
    downloadProvider.Download(targetConfiguration, preDownloadDelegate,
postDownloadDelegate, DownloadOptions.Hardware | DownloadOptions.Software);
}
```

7.19.8.3 Running and stopping PLC

Requirements

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- PLC is offline.

Application

When interacting with TIA Portal through Openness API, it may be necessary to change the operating mode of the PLC. TIA Portal Openness provides a way to modify the operating state of the PLC either to start or stop.

Program code

Modify the following program code for setting PLC operating state to STOP.

```
public void ConfigurePreDownload(DownloadConfiguration downloadConfiguration)
{
    StopModules stopModules = downloadConfiguration as StopModules;
    if (stopModules != null)
    {
        // Puts PLC in "Stop" mode
        stopModules.CurrentSelection = StopModulesSelections.StopAll;
    }
}
```

Modify the following program code for setting PLC operating state to START.

```
public void ConfigurePostDownload(DownloadConfiguration downloadConfiguration)
{
    StartModules startModules = downloadConfiguration as StartModules;
    if (startModules != null)
    {
        // Puts PLC in "Start" mode
        startModules.CurrentSelection = StartModulesSelections.StartModule;
    }
}
```

7.19.8.4 Supporting callbacks

Requirements

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

Certain API methods require an interaction with the user-defined application code during their execution. Delegates are used to handle these callback actions in the user-defined application code. You need to implement a method with a compatible signature, and pass it as a delegate parameter to the action. To proceed with the execution, TIA Portal calls the implemented methods.

Program code

```
// This delegate is declared in Siemens.Engineering.dll
public delegate void
Siemens.Engineering.Download.DownloadConfigurationDelegate(Siemens.Engineering.Download.Co
nfigurations.DownloadConfiguration configuration);
...
Example of an user application code using and implementing the delegate:

[STAThread]
static void Main()
{
    ...
    DownloadProvider downloadProvider = ...;
    IConfiguration targetConfiguration = ...;
    DownloadConfigurationDelegate preDownloadDelegate = PreConfigureDownload;
    DownloadConfigurationDelegate postDownloadDelegate = PostConfigureDownload;
    DownloadResult result = downloadProvider.Download(targetConfiguration,
preDownloadDelegate, postDownloadDelegate, DownloadOptions.Hardware |
DownloadOptions.Software);
    ...
}

//This method will be called back by TIA Portal
private static void ConfigurePreDownload(DownloadConfiguration downloadConfiguration)
{
    // Work with the parameter
}

//This method will be called back by TIA Portal
private static void ConfigurePostDownload(DownloadConfiguration downloadConfiguration)
{
    // Work with the parameter
}
```

Note

STAThread attribute will assure that the delegates are called in the Main thread of execution.

7.19.8.5 Protecting PLC through password

Requirements

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- PLC is offline.

Application

When interacting with TIA Portal through Openness API, it may be necessary to change the protection level of the PLC. TIA Portal Openness provides a way to secure the PLC through a password. The password can be set to both read-protected and write-protected PLCs.

Program code

Modify the following program code for read-protected PLCs

```
public void ConfigurePreDownload(DownloadConfiguration downloadConfiguration)
{
    ModuleReadAccessPassword moduleReadAccessPassword = downloadConfiguration
        asModuleReadAccessPassword;
    if (moduleReadAccessPassword != null)
    {
        SecureString password = ...; // Get password from a secure location
        moduleReadAccessPassword.SetPassword(password); // enter the password to gain
        full access
    }
}
```

Modify the following program code for write-protected PLCs

```
public void ConfigurePreDownload(DownloadConfiguration downloadConfiguration)
{
    ModuleWriteAccessPassword moduleWriteAccessPassword = downloadConfiguration
        asModuleWriteAccessPassword;
    if (moduleWriteAccessPassword != null)
    {
        SecureString password = ...; // Get password from a secure location
        moduleWriteAccessPassword.SetPassword(password); // enter the password to gain full
        access
    }
}
```

 **WARNING**

The API user is responsible for ensuring the security measures of handling passwords through code.

7.19.8.6 Handling PLC block binding passwords

Requirements

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- PLC is offline.

Application

TIA Portal Openness supports the data binding of passwords for customer applications. TIA Portal Openness provides a way for the customer to specify a block binding password. For example, a block binding password can be configured on the DownloadPasswordConfiguration class by calling the SetPassword method.

Note

If you want to secure the download action with a password, a password will have to be provided during every call of download function. This is regardless of whether the device has already been configured. After the successful acceptance of password for a given configuration, all subsequent calls to SetPassword are ignored.

Program code

Modify the following program code:

```
public void ConfigurePreDownload(DownloadConfiguration downloadConfiguration)
{
    DownloadPasswordConfiguration downloadPasswordConfiguration = downloadConfiguration as
    DownloadPasswordConfiguration;
    if(downloadPasswordConfiguration != null &&
    downloadPasswordConfiguration.Message.Contains("block_1"))
    {
        SecureString password = ...; // Get password from a secured location
        downloadPasswordConfiguration.SetPassword(password);
    }
}
```

7.19.9 Functions for accessing PLC service

7.19.9.1 Access level setting

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)

Application

You can use the TIA Portal Openness API to access the Access level setting via the Hardware PlcAccessLevelProvider at the Deviceitem CPU of Device.

This feature provides actions to set / reset the password and a modeled attribute to set / read the access level setting.

- void SetPassword (PlcProtectionAccessLevel accessLevelType, SecureString password)
- void ResetPassword (PlcProtectionAccessLevel accessLevelType)
- attribute PlcProtectionAccessLevel

In general the handling via Openness is similar to the TIA Portal UI.

- PlcProtectionAccessLevel can always be read and set at S7-1200/1500 PLCs.
- If access level is set to anything different than Full access (or Full access incl. fail-safe) a full access password has to be set. Otherwise there will be a compile error: "Password must not be empty"

- It is not allowed to set the same password for different access levels
- It is only possible to set / reset passwords for higher levels than the one set as access level (if PlcProtectionAccessLevel is set to HMIAccess it is only possible to set a password for ReadAccess and FullAccess)

Possible exceptions:

- EngineeringTargetInvocationException: "**Same password already exists for other access level**" - will be thrown if the user tries to set the same password for different access levels
- EngineeringTargetInvocationException: "**Cannot set password for the access level**" - will be thrown if the user tries to set a password for a stricter access level than the one selected
- EngineeringTargetInvocationException: "**Password cannot be empty**" - will be thrown if the user tries to set an empty password
- EngineeringTargetInvocationException: "**Access Level is not valid**" - will be thrown if the user tries to set the access level to FullAccessIncludingFailsafe at non failsafe PLCs

Access level

In Openness the access level is modelled as an Enum named PlcProtectionAccessLevel. The name of the Enum fields are based on the "Access Level" entries for S7-1200/1500 PLCs.

TIA UI name	Enum entry	Value	Remarks
-	None	0	For enum initialization. This value must not be set by the Openness user.
Full access (no protection)	FullAccess	1	
Read access	ReadAccess	2	
HMI access	HMIAccess	3	
No access	NoAccess	4	
Full access incl. fail-safe (no protection)	FullAccessIncludingFailsafe	5	Only available at failsafe PLCs

An EOM Attribute(Modeled Attribute) named PlcProtectionAccessLevel of type PlcProtectionAccessLevel enum is available at PLCs to Set/Get the access level.

You can use the following code example for PlcProtectionAccessLevel with No access:

```
DeviceItem S71500PLC = ....;
PlcAccessLevelProvider myPlcAccessLevelProvider =
S71500PLC.GetService<PlcAccessLevelProvider>();
myPlcAccessLevelProvider.PlcProtectionAccessLevel = PlcProtectionAccessLevel.NoAccess;
```

Password

The password is implemented as SecureString for security reasons. It is only possible to set / reset the passwords for the different access levels using the corresponding action. Reading the passwords is not supported.

```
myPlcAccessLevelProvider.SetPassword(PlcProtectionAccessLevel.ReadAccess,  
someSecureString);  
myPlcAccessLevelProvider.ResetPassword(PlcProtectionAccessLevel.ReadAccess);
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.19.9.2 Accessing OPC UA server interface

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open
See [Opening a project \(Page 106\)](#)

Application

You can create server interface objects and import XML files that define the OPC UA server interface for the PLC.

You can use the TIA Portal Openness API to access the following functionality using OpcUaProvider service:

- Adding an OPC UA server interface object
- Enumerating through the server interface belonging to PLC
- Importing a server interface file
- Exporting an existing server interface to XML
- Enabling/ Disabling server interface
- Deleting a server interface

Program code

You can interact with OPC UA server interfaces via an OpcProvider service that can be acquired from PlcSoftware.

```
PlcSoftware software = ....;
OpcUaProvider provider = software.GetService<OpcUaProvider>();
if (provider != null)
{
    ....
}
```

Modify the following program code to allow navigation to the OPC UA server interfaces:

```
PlcSoftware software = ....;
OpcUaProvider provider = software.GetService<OpcUaProvider>();
if (provider != null)
{
    OpcUaCommunicationGroup commGroup = provider.CommunicationGroup;
    ServerInterfaceGroup serverInterfaceGroup = commFolder.ServerInterfaceGroup;
    ServerInterfaceComposition serverInterfaces = serverInterfaceGroup.ServerInterfaces;
}
```

Modify the following program code to enumerate through the items in the ServiceInterfaceComposition collection:

```
ServerInterfaceComposition serverInterfaces = ....;
foreach (ServerInterface serverInterface in serverInterfaces)
{
    ....;
}
```

Modify the following program code to add an OPC UA server interface object:

```
ServerInterfaceComposition serverInterfaces = ....;
if (serverInterfaces != null)
{
    ServerInterface serverInterface = serverInterfaces.Create(name);
}
```

Modify the following program code to get and set the properties of OPC UA server interface object:

```
ServerInterfaceComposition serverInterfaces = ....;
if (serverInterfaces != null)
{
    ServerInterface serverInterface = serverInterfaces.Create("New Interface");
    serverInterface.Author = "James Cornett";
}
```

Modify the following program code to set the contents of the server interface object based on XML file:

```
ServerInterfaceComposition serverInterfaces = ....;
if (serverInterfaces != null)
{
ServerInterface serverInterface = serverInterfaces.Create(name);
if (serverInterface != null)
{
serverInterface.Import(new FileInfo(importFilePath));
}
}
```

Modify the following program code to write the contents of the server interface object to an XML file:

```
String exportFilePath = Path.Combine(Directory.GetCurrentDirectory(),
"ExportInterface.xml");
ServerInterface serverInterface = ...;
serverInterface.Export(new FileInfo(exportFilePath));
```

Modify the following program code to remove the object from the containing ServerInterfaceComposition, and de-allocates the object:

```
ServerInterface serverInterface = ...;
serverInterface.Delete();
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.19.9.3 Accessing Software Checksum

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- You have opened a project with your TIA Portal Openness application
See [Opening a project \(Page 106\)](#)

Application

You can use the Openness API to access the software checksum for a PLC station.

Program code

To access, you get the instance of PlcSoftware and load a service instance of PlcChecksumProvider.

```
PlcSoftware plc = ...;
//get PlcSoftware instance
PlcChecksumProvider checksumProvider =
plc.GetService<PlcChecksumProvider>();
```

In case the PLC does not support the checksum calculation then the GetService call returns null.

At the PlcChecksumProvider instance the software checksum can be reached as follow:

```
string softwareChecksum = checksumProvider.Software;
```

The Software attribute is read only and it returns null when the program is not compiled.

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.19.9.4 Assigning PC interface

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open
See [Opening a Project \(Page 106\)](#)

Application

For the interface assignment, you have to get a PciInterfaceAssignment service from Siemens.HW.Features on the device item interface.

If the PC-Station version is <= 2.0, the methods of this service will return null and interface assignment will not be possible for the interfaces that are assigned to PC-Station or Windows only or none. For the interfaces that are assigned to SW-CPU, this is again possible.

Note

The support of PC Station is very limited in the CAx interchange. Only PC Stations with version 1.0 and the default value for InterfaceAssignment are supported. The reason is these two values have impact on device structure (what is built-in, what can be plugged where). But since the parameters are not part of the AutomationML file, the values are not set during import and as a result the structure of the IPC does not fit and the other modules can not be plugged.

Program code: Interface assignment

Modify and use the following program code for interface assignment:

```
DeviceItem swCpu = ...;
DeviceItem myInterface = ...;
PcInterfaceAssignment provider = myInterface.GetService<PcInterfaceAssignment>();
//user has to give device item sw cpu in case he/she wants to assign to it. (in the future,
there could be multi sw-cpu cases)
provider.AssignInterface(PcInterfaceAssignmentMode.SoftwarePlc, swCpu);
...
provider.AssignInterface(PcInterfaceAssignmentMode.PcStation);
...
provider.AssignInterface(PcInterfaceAssignmentMode.None);
...
```

Modify and use the following program code to get information about the current assignment mode of an interface:

```
PcInterfaceAssignment provider = myInterface.GetService<PcInterfaceAssignment>();
PcInterfaceAssignmentMode mode = provider.PcInterfaceAssignmentMode;
switch(mode)
{
    case PcInterfaceAssignmentMode.None: Console.WriteLine("Assigned to none or windows-
only.");
    break;
    case PcInterfaceAssignmentMode.PcStation: Console.WriteLine("Assigned to pc-station.");
    break;
    case PcInterfaceAssignmentMode.SoftwarePlc: Console.WriteLine("Assigned to SoftwarePlc: " +
+ provider.SoftwarePlc.Name);
    break;
}
```

Program code: Hardware Resource and IPC Expansion configuration

You can also select correct IPC Expansion and HardwareResource property via `PcInterfaceAssignment` service using the following code:

```
DeviceItem myInterface = ...;
PcInterfaceAssignment provider = myInterface.GetService<PcInterfaceAssignment>();
//this method returns a subset of IpcExpansion values which are available for the IPC that
the interface is plugged in. IEnumerable<string> ipcExpansionChoices =
provider.GetAvailableIPCExpansions ();
string myChoice;
foreach(var choice in ipcExpansionChoices)
{
//user must select the desired value depending on his/her configuration
if (choice.Contains("3yxx1"))
{
    myChoice = choice;
    break;
}
}
provider.IpcExpansion = myChoice;
//after assigning IpcExpansion value, user may assign HardwareResource value using the
following enumeration. provider.HardwareResource = HardwareResource.X101;
//OR
//myInterface.SetAttribute("HardwareResource", HardwareResource.X101);
//myInterface.SetAttribute("IpcExpansion", mychoice);
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.19.10 Uploading hardware, software and files to PLC device

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
See [AUTOHOTSPOT](#)
- You have opened a project with your TIA Portal Openness application.
See [AUTOHOTSPOT](#)

Application

Openness user is able to upload station into a project through `StationUploadProvider` (accessed from a given project). An upload into a `DeviceGroup` is not supported. If a project is used to execute an upload, an instance of `StationUploadProvider` will be returned on `GetService` call, else the service will return null.

Program code: Retrieving StationUploadProvider service from a project

```
Project myProject = ...;
StationUploadProvider uploadProviderForProject =
myProject.GetService<StationUploadProvider>();
if (uploadProviderForProject != null)
{
    ...
}
```

Parameters of upload method

In order to execute an upload of a PLC device, user calls StationUpload method of StationUploadProvider. The Upload method has ConfigurationAddress and UploadConfigurationDelegate parameters. UploadOptions are optional, because the StationUpload uploads Software, Hardware, and Files.

Parameter name	Type	Description
configurationAddress	Siemens.Engineering.Connection.ConfigurationAddress	Address of device that should be uploaded
uploadConfigurationDelegate	Siemens.Engineering.Upload.UploadConfigurationDelegate	Delegate that will be called to check configuration before upload
uploadOptions	Siemens.Engineering.Upload.UploadOptions	Upload options

Parameter 1: ConfigurationAddress

The user should provide ConfigurationAddress object to the Upload. The address object is used to establish a connection to the given PLC device that should be uploaded. The ConfigurationAddress object must be created in the ConnectionConfiguration of the StationUploadProvider. The Configuration contains a list of supported Modes. You need to select one of the Modes that should be used for upload. The selected ConfigurationMode contains a list of all local PclInterfaces that support the selected Mode, you have to select one of the interfaces. The desired address can be created in the Address collection of the selected ConfigurationPclInterface.

Modify the following code to create an address object:

```
...
StationUploadProvider uploadProvider = null;
...
ConnectionConfiguration configuration = uploadProvider.Configuration;
ConfigurationMode configurationMode = configuration.Modes.Find("PN/IE");
ConfigurationPcInterface pcInterface = configurationMode.PcInterfaces.Find("Intel(R)
Ethernet Connection I217-LM", 1);
//Create an address. This "ConfigurationAddress" is used as parameter for upload."
ConfigurationAddress uploadAddress = pcInterface.Addresses.Create("192.68.0.1");
...
```

Project upload

The user can start the station upload by calling the action StationUpload.

The following Parameters are mandatory:

- ConfigurationAddress: The address of the device to be uploaded
- UploadConfigurationDelegate: The callback to handle upload inhibits

```
...
StationUploadProvider uploadProvider = null;
Device uploadedObject = null;
...
UploadConfigurationDelegate preUploadDelegate = PreConfigureUpload;
UploadResult result = uploadProvider.StationUpload(uploadAddress, preUploadDelegate);
// The uploaded device
uploadedObject = resultUploadedStation;
if (uploadedObject == null)
{
    ...
}
internal void PreConfigureUpload(UploadConfiguration uploadConfiguration)
{
    ...
}
```

Parameter2: UploadConfigurationDelegate

Openness user needs to provide an implementation of `void UploadConfigurationDelegate (UploadConfiguration uploadConfiguration)`. The delegate will be called for pre-upload configurations. The delegate will be called for each configuration that requires an action from the user. For more information about callback handling, Refer AUTOHOTSPOT. Certain configurations will only contain an information, therefore the user action will not be required.

The possible upload configuration types are listed below:

Configuration name	Description and properties
UploadConfiguration	<ul style="list-style-type: none"> • Base class for all the configurations. It contains information in the Message attribute • Contains single property UploadConfiguration.Message : string (read only property contains the configuration message)
UploadPasswordConfiguration	<ul style="list-style-type: none"> • Derived from UploadConfiguration • Base class for all configuration that required a password for upload. • Contains a single method to set password. UploadPasswordConfiguration.SetPassword (password: SecureString) : void - Set password
UploadSelectionConfiguration	<ul style="list-style-type: none"> • Derived class of UploadConfiguration • Does not contain additional properties

The datatype of the configurations are given below:

Configuration	DataType	Description and Action
UploadPasswordConfiguration	ModuleReadAccessPassword	<p>Set password via SetPassword(password:SecureString) method. Enter a password to gain read access to the module.</p>
	PasswordReadAccess	<p>Set password via SetPassword(password:SecureString) method. Enter a password for SW Upload in classic PLC's to gain read access to the module.</p>
UploadSelectionConfiguration	UploadMissingProducts	<p>Set CurrentSelection:UploadMissingProductsSelections Available enum values: TryUpload (Consistent upload) NoAction (No action) Set a selection for upload.</p>

The support of a Failsafe password is not necessary. For the read-access by uploading a F-PLC no password is needed.

Unhandled configuration that can prevent the upload causes an EngineeringTargetInvocationException and aborts upload.

An EngineeringDelegateInvocationException will be thrown in case of an unhandled exception within the Delegate.

PreUploadDelegate implementation example:

```

private static void PreConfigureUpload(UploadConfiguration UploadConfiguration)
{
    ModuleReadAccessPassword moduleReadAccessPassword = UploadConfiguration as
ModuleReadAccessPassword;
    if (moduleReadAccessPassword != null)
    {
        string passWD = "passWD";
        var password = new SecureString();
        foreach (var c in passWD)
            password.AppendChar(c);

        moduleReadAccessPassword.SetPassword(password);
        return;
    }

    ModuleWriteAccessPassword moduleWriteAccessPassword = UploadConfiguration as
ModuleWriteAccessPassword;
    if (moduleWriteAccessPassword != null)
    {
        string passWD = "passWD";
        var password = new SecureString();
        foreach (var c in passWD)
            password.AppendChar(c);

        moduleWriteAccessPassword.SetPassword(password);
        return;
    }
    ...

    throw new NotSupportedException(); // Exception thrown in the delegate will cancel upload
}

```

Parameter3: UploadOptions

The user cannot specify the Upload options. This Upload options are known as: "Hardware", "Software", "Hardware and Software" and "Hardware, Software and Files".

UploadResult

The UploadResult returned by the Upload action provides feedback on the state of the objects that were uploaded.

- UploadResult.Message: UploadResultMessageComposition - Composition of UploadResultMessage

The following attributes are supported:

Attributes	Description
ErrorCount	int value of errors while upload
State	UploadResultState with possibly values: Success, Information, Warning and Error

Attributes	Description
UploadedStation	A Device-Instance of the uploaded station
WarningCount	Number of warning while upload as int

The UploadResultMessage contains:

- UploadResultMessage.Messages : UploadResultMessageComposition - Composition of UploadResultMessage

The following attributes are supported:

Attributes	Description
DateTime	System.DateTime of the created message.
ErrorCount	An int counter for errors.
State	UploadResultState with possibly values: Success, Information, Warning and Error.
WarningCount	Number of warning while upload as int

Upload invocation example

```
internal bool UploadPLC()
{
    ...
    UploadResult result = uploadProvider.StationUpload(uploadAddress, preUploadDelegate);
    ...
    PrintAllMessages(result.Messages, 0);
    ...
}

internal void PrintAllMessages(UploadResultMessageComposition messages, int level)
{
    if (messages == null)
        return;

    if (level == 0)
        Console.WriteLine("\n");
    foreach (UploadResultMessage message in messages)
    {
        string messageOut = message.Message.PadLeft(message.Message.Length + level, '\t') + "\n";
        Console.WriteLine(messageOut);

        if ((message.Messages != null) && (message.Messages.Count > 0))
            PrintAllMessages(message.Messages, level+1);
    }
}
```

7.19.11 Comparing PLC software

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See AUTOHOTSPOT
- You have opened a project with your TIA Portal Openness application.
See AUTOHOTSPOT

Application

You have the following options to determine the deviation between the software of two devices:

- Comparing the software of two configured PLCs
- Comparison of the software of a PLC and the project library
- Comparison of the software of a PLC and the global library
- Comparison of the software of a PLC and the master copy of a PLC
- Comparison of the software of a configured PLC with the software of a connected PLC in "Online" status

Signature

Use the CompareTo or CompareToOnline methods for the comparison.

```
public CompareResult CompareTo (ISoftwareCompareTarget compareTarget)
```

```
public CompareResult CompareToOnline ()
```

Return value / parameter	Function
CompareResult compareResult	Returns the comparison result: <ul style="list-style-type: none"> • FolderContentsDifferent: Content of the compared folders differs. • FolderContentsIdentical: Content of the compared folders is identical. • ObjectsDifferent: Content of the compared objects differs. • ObjectsIdentical: Content of the compared objects is identical. • LeftMissing: The object is not contained in the object from which the comparison was started. • RightMissing: The object is not contained in the object which is being compared. • CompareIrrelevant: Comparison of these 2 objects is irrelevant • FolderContainsDifferencesOwnStateDifferent: Folder contents have one or more differences, folder's own state different • FolderContentEqualOwnStateDifferent: Folder content is the same, folder's own state is different
ISoftwareCompareTarget compareTarget	List of comparable objects.

Program code

Modify the following program code to output the comparison result:

```
private static void WriteResult(CompareResultElement compareResultElement, string indent)
{
    Console.WriteLine("{0} <{1}> <{2}> <{3}> <{4}> ",
        indent,
        compareResultElement.LeftName,
        compareResultElement.ComparisonResult,
        compareResultElement.RightName,
        compareResultElement.DetailedInformation);
    WriteResult(compareResultElement.Elements, indent);
}
private static void WriteResult (IEnumerable<CompareResultElement> compareResultElements,
string indent)
{
    indent += " ";
    foreach (CompareResultElement compareResultElement in compareResultElements)
    {
        WriteResult(compareResultElement, indent);
    }
}
```

7.19 Functions for accessing the data of a PLC device

Modify the following program code to compare the software of devices:

```
private static void CompareTwoOfflinePlcs(PlcSoftware plcSoftware0, PlcSoftware
plcSoftware1)
{
    if (plcSoftware0 != null && plcSoftware1 != null)
    {
        CompareResult compareResult = plcSoftware0.CompareTo(plcSoftware1);
        WriteResult(compareResult.RootElement, string.Empty);
    }
}
```

Modify the following program code to compare the software of a PLC with the project library:

```
private static void ComparePlcToProjectLibrary(Project project, PlcSoftware plcSoftware)
{
    if (project != null && plcSoftware != null)
    {
        CompareResult compareResult = plcSoftware.CompareTo(project.ProjectLibrary);
        WriteResult(compareResult.RootElement, string.Empty);
    }
}
```

Modify the following program code to compare the software of a PLC with the global library:

```
private static void ComparePlcToGlobalLibrary(PlcSoftware plcSoftware, GlobalLibrary
globalLibrary)
{
    if (plcSoftware != null && globalLibrary != null)
    {
        CompareResult compareResult = plcSoftware.CompareTo(globalLibrary);
        WriteResult(compareResult.RootElement, String.Empty);
    }
}
```

Modify the following program code to compare the software of a PLC with a master copy:

```
private static void ComparePlcToMasterCopy(Project project, PlcSoftware plcSoftware)
{
    if (project != null && plcSoftware != null)
    {
        CompareResult compareResult =
plcSoftware.CompareTo(project.ProjectLibrary.MasterCopyFolder.MasterCopies[0]);
        WriteResult(compareResult.RootElement, string.Empty);
    }
}
```

Modify the following program code to compare the software of a PLC with the software of a connected PLC:

```
private static void ComparePlcToOnlinePlc(PlcSoftware plcSoftware)
{
    if (plcSoftware != null)
    {
        CompareResult compareResult = plcSoftware.CompareToOnline();
        WriteResult(compareResult.RootElement, string.Empty);
    }
}
```

7.19.12 Comparing PLC hardware

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal
- You have opened a project with your TIA Portal Openness application.
See Opening a project

Application

You can use the TIA Portal Openness to compare the hardware of two PLC devices.

Signature

Use the CompareTo method for the comparison of two hardware objects.

CompareResult CompareTo (IHardwareCompareTarget compareTarget);

Return value/parameter	Function
CompareResult compare result	Return the comparision result: <ul style="list-style-type: none"> • FolderContainsDifferencesOwnStateDifferent: Folder contents have one or more differences, folder's own state is different • FolderContentEqualOwnStateDifferent: Folder content is the same, folder's own state is different.
IHardwareCompareTarget compareTarget	The compare target for which the hardware compare should be performed. Must not be null.

If the Parameter compareTarget is null and an attempt is made to compare the hardware will throw Siemens.Enginnering.EngineeringTargetInvocationExceptions.

Program

Modify the following program code to output the comparison result:

```
...
CompareResult compareResult = plc_1.CompareTo(plc_2);
CompareResultState resultState = compareResult.RootElement.ComparisonResult;
if (resultState == CompareResultState.FolderContainsDifferencesOwnStateDifferent)
{
    // Folder contents have one or more differences, folder's own state is different:
    // May occur if the plc has a different subordinate element, e.g., a local module, and
    // the plc itself is different, e.g., in a parameter
}
else if (resultState == CompareResultState.FolderContentEqualOwnStateDifferent)
{
    // Folder content is the same, folder's own state is different:
    // May occur if a folder-style module, e.g., FM 351, has equal subordinate elements but
    // the module itself is different, e.g., in a parameter
}
else if (resultState == CompareResultState.FolderContentsIdentical)
{
    ...
}
...
...
```

7.19.13 Establishing or disconnecting the online connection to the PLC

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See AUTOHOTSPOT
- A project is open.
See AUTOHOTSPOT
- All devices are enumerated.
See AUTOHOTSPOT.

Application

You can establish the online connection to a PLC, or disconnect an existing online connection.

Program code

Modify the following program code to establish or disconnect the online connection to a PLC:

```
public static void SetOnlineConnection(DeviceItem deviceItem)
{
    OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
    if (onlineProvider == null) { return; }
    // Go online
    if (onlineProvider.Configuration.IsConfigured)
    {
        onlineProvider.GoOnline();
    }
    // Go offline
    onlineProvider.GoOffline();
}
```

You can also establish or disconnect the online connections to all available PLCs in a project.

```
public static void SetOnlineConnectionForAllPLCs(Project project)
{
    foreach (Device device in project.Devices)
    {
        foreach (DeviceItem deviceItem in device.DeviceItems)
        {
            OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
            if (onlineProvider != null)
            {
                // Establish online connection to PLC:
                onlineProvider.GoOnline();

                // ...

                // Disconnect online connection to PLC:
                onlineProvider.GoOffline();
            }
        }
    }
}
```

7.19.14 Assigning project language to PLC

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal
- A project is open
See Opening a Project

Application

You can use the TIA Portal Openness to assign project languages to web server and display languages of an S71500 PLC. To achieve that a StructuredData type dynamic attributes is required.

To access multilingual settings the new dynamic attribute MultilingualSupport of type TableData has been added. This can be split up into Rows. At each Row the assigned project language can be set or read with the dynamic attribute ProjectLanguage.

Program code

Modify the following program code to assign project language to PLC:

```
//To Set the Project Language
LanguageSettings languageSettings = project.LanguageSettings;
LanguageAssociation activeLanguages = languageSettings.ActiveLanguages;
DeviceItem Plc = ...;
Tabledata multilingualSupportTable = Plc.GetAttribute("MultilingualSupport");
StructuredDataComposition structuredDataComp = multilingualSupportTable.Rows; // This will
return a collection of Structured Data.
StrcturedData firstRow= structuredDataComp[0]; //First row
Language activeGermanLanguage = activeLanguages.Find(CultureInfo.GetCultureInfo("de-DE"));
firstRow.SetAttribute("ProjectLanguage", activeGermanLanguage.Culture);
//To Get the Languages of device displayed
var assignedToGerman = firstRow.GetAttribute("DisplayLanguage");
```

7.19.15 Assigning watch & force tables for web server and PLC display

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal
- A project is open
See Opening a Project

Application

You can use the TIA Portal Openness to assign already created watch and force tables to the web server. The watch table and force table are software that can be found, created, and deleted in the software container. You can use the TIA Portal Openness to export/ import Watch and Force tables. For export/import Watch/Force tables, please see Export/Import Watch & Force Table

To assign the watch and force tables to the web server, the WatchAndForceTableAccessManager service is used at the PLC DeviceItem.

The service contains two navigators for watch and force tables.

- WatchtableAccessRules {WatchtableAccessRuleComposition}
- ForcetableAccessRules {ForcetableAccessRuleComposition}

The navigator WatchTableAccessRules provides the WatchTableAccessRuleComposition which contains objects of type WatchTableAccessRule. The composition is empty by default. For the WatchTableAccessRule objects, the actions Find and Create are defined.

WatchTable Access Rule & ForceTable AccessRule

- **PlcWatchTable** watchtable {get;} returns the software object watchtable. In order to exchange an assigned watch table, the user has to remove the assignment of the current one (using WatchTableAccessRule.Delete()) and add a new one (using WatchTableAccessRule.Create()).
- **WatchAndForceTableAccess** access {get; set;} returns / sets the web server access level whether it is read or read/write

TIA UI name	value	Openness enum description
-	0	None
Read	1	Read
Read/Write	2	Write

Error will be thrown in the following cases:

- If the web server has not been activated (WebserverActivate == FALSE), a EngineeringTargetInvocationException is thrown with error details stating that a WatchTableAccessRule can be added only if the web server is enabled.
- If the user tries to add a watch table with WatchAndForceTableAccess.None a ConfigOpennessUserException with error details stating that WatchAndForceTableAccess.None is not allowed as a access level

Program code: Assigning & unassigning watch table

Modify the following program code to search watch table in the web server:

```
WatchAndForceTableAccessManager mngr =
deviceItem.GetService<WatchAndForceTableAccessManager>();
WatchTableAccessRuleComposition watchTableCmp = mngr.WatchTableAccessRules;
WatchTableAccessRule accessRule1 = watchTableCmp.Find(watchTable1);
```

Note

Null will be returned if that watchtable is not associated with any WatchTableAccessRule in the Webserver

Modify the following program code to create a new watch table to the web server with read access:

```
WatchAndForceTableAccessManager mngr =
deviceItem.GetService<WatchAndForceTableAccessManager>();
WatchTableAccessRuleComposition watchTableCmp = mngr.WatchTableAccessRules;
watchTableCmp.Create(watchTable1, WatchAndForceTableAccess.Read);
```

Modify the following program code to delete an existing WatchTableAccessRule and unassign the watch table from the web server:

```
WatchTableAccessRule whatchtable= watchTableCmp.Find(watchTable1);
whatchtable.Delete();
```

Error will be thrown in the following cases:

- If the web server has not been activated (WebserverActivate == FALSE), a EngineeringTargetInvocationException is thrown with error details stating that a WatchTableAccessRule can be added only if the web server is enabled
- If a WatchTableAccessRule with the watch table already exists, a ConfigOpenessUserException is thrown with error details stating that the watchtable already exists.
- If you try to add a watch table with WatchAndForceTableAccess.None, a ConfigOpenessUserException with error details stating that WatchAndForceTableAccess.None is not allowed as a access level.

Program code: Assigning force table to the web server

Modify the following program code to search force table in the web server:

```
WatchAndForceTableAccessManager mngr =
deviceItem.GetService<WatchAndForceTableAccessManager>();
ForceTableAccessRuleComposition forceTableCmp = mngr.ForceTableAccessRules;
ForceTableAccess forceTable = forceTableCmp.Find(forceTable1);
```

Modify the following program code to create PLC force table to the web server with read access:

```
ForceTableAccessRuleComposition forceTableCmp = mngr.ForceTableAccessRules;
forceTableCmp.Create(forceTable1, WatchAndForceTableAccess.Read);
```

Watch and force tables at the PLC display

You can use the same Openness service WatchAndForceTableAccessManager available at the Display submodule (DeviceItem) of the PLC DeviceItem. The same web server functionality is used as described above to assign watch and force tables at the PLC display. In contrast to the web server, the display cannot be disabled, so here a similar validation as for WebserverActive is not available.

7.19.16 Accessing web server and OPC UA user management

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal
- A project is open
See Opening a Project

Application

You can use the TIA Portal Openness to access web server and OPC UA submodule of PLCs. It is possible to add a user up to maximum number at the web server and the OPC UA submodule of PLCs. The user has a user name and a password.

User management at Web Server

You can perform the following operations, such as adding user, deleting user, and setting password at web server for all supported device only if web server is activated at the module.

If the web sever is not activated, the add, delete and set operations will throw a `EngineeringTargetInvocationException`, but still the read operations will be available.

The `WebServerUserManagement` service will be available for the PLC instance of only the supported devices. For other non-supported devices, the service will be null.

```
WebServerUserManagement
{
    Navigators WebServerUsers
    {
        WebServerUserComposition
    }
}
```

The Service provides a navigator called `WebServerUsers` using which a `WebServerUserComposition` can be obtained, which manages the `WebServerUser` instances.

```
WebServerUserComposition
{
    WebServerUser Find(string username);
    void Create(string username, WebserverUserPermissions permissions, SecureString password);
}
WebServerUser
{
    string UserName{get;}
    WebserverUserPermissions Permissions {get;set;}
    void Delete();
    void SetPassword(SecureString password);
}
```

Program code: Actions at WebServerUserComposition

Modify the following program to search web server user:

```
WebServerUserComposition webServerUserComposition = WebServerUserManagement.WebServerUsers;
WebServerUser user1 = webServerUserComposition.Find("user1");
```

Modify the following program code to create a new web server user with a web server user permissions:

```
WebServerUserComposition webServerUserComposition = WebServerUserManagement.WebServerUsers;
WebServerUser user1 = webServerUserComposition.Create("user1",
WebserverUserPermissions.ReadTagStatus, someSecureString);
```

Modify the following program code to delete the web server user:

```
WebServerUserComposition webServerUserComposition = WebServerUserManagement.WebServerUsers;
WebServerUser user1 = webServerUserComposition.Find("user1");
user1.Delete();
```

Modify the following program code to replace the current password of the user with the new password:

```
WebServerUserComposition webServerUserComposition = WebServerUserManagement.WebServerUsers;
WebServerUser user1 = webServerUserComposition.Find("user1");
user1.SetPassword(someSecureString);
```

User Management at OPC UA

You can perform the following operations such as adding user, deleting user, and setting password at the OPC UA Server only if the OPC UA Server is activated and username and password authentication is enabled.

If these conditions are not met, the operation throws a `EngineeringTargetInvocationException` with error details stating that the authentication needs to be enabled.

The `OpcUaUserManagement` service will be available on the OPC UA submodule of the PLC. The Service will be available for all supported devices where OPC UA submodule is available for the device item. In other cases, the service will be returned as null.

```
OpcUaUserManagement
{
    Navigators OpcUaUsers
    {
        OpcUaUserComposition
    }
}
```

The Service provides a navigator called OpcUaUsers using which a OpcUaUserComposition can be obtained. The composition manages the OpcUaUser instances:

```
OpcUaUserComposition
{
OpcUaUser Find(string username);
void Create(string username, SecureString password);
}
OpcUaUser
{
string UserName{get;}
void Delete();
void SetPassword(SecureString password);
}
```

Program code: Actions at OpcUaUserComposition

Modify the following program code to search a OpcUaUser:

```
OpcUaUserComposition opcUaUserComposition = opcUaUserManagement.OpcUaUsers;
OpcUaUser user1 = opcUaUserComposition.Find("user1");
```

Modify the following program code to create a new OpcUaUser:

```
OpcUaUserComposition opcUaUserComposition = opcUaUserManagement.OpcUaUsers;
opcUaUserComposition.Create("user1", someSecureString);
```

Modify the following program code to delete the OpcUaUser:

```
OpcUaUserComposition opcUaUserComposition = opcUaUserManagement.OpcUaUsers;
OpcUaUser user = opcUaUserComposition.Find("user1");
user.Delete();
```

Modify and use the program code to replace the current password with a supplied one:

```
OpcUaUserComposition opcUaUserComposition = opcUaUserManagement.OpcUaUsers;
OpcUaUser user = opcUaUserComposition.Find("user1");
user.SetPassword(someSecureString);
```

7.19.17 Accessing domain settings

Requirements

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal
- A project is open
See Opening a project

Introduction

You can use the TIA Portal Openness to access the complete domain management. The following two new services are added to the Subnet class for accessing domain management via TIA Portal Openness:

- service MrpDomainOwner
- service SyncDomainOwner

Program code: Accessing MrpDomainOwner

The service MrpDomain provides the navigator MrpDomains which contains the MrpDomainComposition. The composition contains objects of type MrpDomain.

```
Subnet subnet = ...;
MrpDomainOwner mrpDomainOwner = subnet.GetService<MrpDomainOwner>();
MrpDomainComposition mrpDomainComposition = mrpDomainOwner.MrpDomains;
```

Modify the following program code to create a new domain with a name newMrpDomain;

```
int countOfMrpDomains = mrpDomainComposition.Count;
MrpDomain someMrpDomain = mrpDomainComposition.ElementAt(3);
MrpDomain firstMrpDomain = mrpDomainComposition.First();
MrpDomain byName = mrpDomainComposition.Find("DomainName");
// create a new domain
MrpDomain newMrpDomain = mrpDomainOwner.MrpDomains.Create("newMrpDomain");
```

Modify the following program code to read/write attributes value of a MRP domain objects:

```
NetworkInterface toBeAdded = ...;
newMrpDomain.SetAttribute("IsDefault", true);
newMrpDomain.SetAttribute("ManagerOutsideOfProjectActive", false);
var participants = firstMrpDomain.DomainParticipants;
int count = participants.Count;
participants.Add(toBeAdded);
foreach (NetworkInterface networkIf in participants)
{
// do something at the interface
}
newMrpDomain.Delete();
```

Program code: Accessing SyncDomainOwner

The service provides the navigator SyncDomains which contains the SyncDomainComposition. This composition contains objects of type SyncDomain.

```
Subnet subnet = ...;
SyncDomainOwner syncDomainOwner = subnet.GetService<SyncDomainOwner>();
SyncDomainComposition syncDomainComposition = syncDomainOwner.SyncDomains;
```

Modify the following program code to create a new SyncDomain with specific name:

```
int countOfSyncDomains = syncDomainComposition.Count;
SyncDomain someSyncDomain = syncDomainComposition.ElementAt(3);
SyncDomain firstSyncDomain = syncDomainComposition.First();
SyncDomain byName = syncDomainComposition.Find("DomainName");
// create a new domain
SyncDomain newSyncDomain = syncDomainOwner.SyncDomains.Create("newSyncDomain");
```

Modify the following program code to read/write the attribute value of Sync Domain:

```
NetworkInterface toBeAdded = ...;
string convertedName = newSyncDomain.ConvertedName;
newSyncDomain.SetAttribute("IsDefault", true);
newSyncDomain.SetAttribute("HighPerformanceActive", true);
newSyncDomain.SetAttribute("FastForwardingActive", true);
var participants = firstSyncDomain.DomainParticipants;
int count = participants.Count;
participants.Add(toBeAdded);
foreach (NetworkInterface networkIf in participants)
{
    // do something at the interface
}
newSyncDomain.Delete();
```

7.19.18 Setting a display password

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal
- A project is open
See Opening a Project

Application

You can use the TIA Portal Openness to set the display protection password. You can use the method SetPassword(SecureString password) to set the display protection password where the parameter for the password to be a SecureString. The confirmation password is not needed in Openness and therefore not available.

The feature will be available if the following preconditions are met.

- The PLC has a “Display” instance.
- The “Display” supports “Password” protection feature. There are cases eg. Software PLC where Display is available but no password protection available.

Program code

Modify the following program code to get the device item that owns the instance of the service, the property OwnedBy can be used.

```
DisplayProtection displayProtection = plcDisplay.GetService<DisplayProtection>();  
var displayDeviceItem = displayProtection.OwnedBy;
```

Modify the following program code to set the Display Protection password:

```
displayProtection.SetPassword(someSecuredString);  
//Sets the string someSecuredString as the CPU display protection password.
```

Note

SW controllers have a display submodule, but do not support any display protection. So in TIA Portal Openness the feature SetPassword is not available at the display submodule of any SW controllers.

7.19.19 Supporting IP Accessibility

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal
- A project is open
See Opening a project

Introduction

You can use the TIA Portal Openness to support IP accessibility. In TIA Portal Openness, a new dynamic attribute PlcAccessCommunicationModule gets available at the device item CPU. The attribute is of type object, so that you can access to the CP object you want to select for IP Accessibility and assign the object to PlcAccessCommunicationModule.

If a CP has already been selected you can retrieve the CP object by getting the value of PlcAccessCommunicationModule.

Program code

```
DeviceItem Plc = ...;
object communicationModule = Plc.GetAttribute("PlcAccessCommunicationModule");
// cP contains the object reference of a plugged CP
Plc.SetAttribute("PlcAccessCommunicationModule", cP);
```

7.19.20 Updating module description

Requirement

- The TIA Portal Openness is connected to the TIA Portal
See Connecting to the TIA Portal
- A project is open
See AUTOHOTSPOT

Application

In TIA Portal Openness, a specific service ModuleDescriptionUpdater is used at the device item to update the current module description to the newest version of the module. To get the device item that owns the instance of the service, the property OwnedBy can be used.

You can use the CanUpdate attribute to show if a new ConfigObject version for this deviceitem is available.

Atribute	Data type	Description
CanUpdate	bool	TRUE: A new version is available FALSE: There is no new version

7.19 Functions for accessing the data of a PLC device

You can use the action `UpdateModuleDescription()` to update the `ConfigObject` version of a `deviceitem`.

Action	Return value	Description
UpdateModuleDescription	True	The <code>deviceitem</code> is up to date: <ul style="list-style-type: none"> • The <code>deviceitem</code> was updated successfully • The <code>deviceitem</code> was already up to date
	False	The <code>deviceitem</code> is not up to date: <ul style="list-style-type: none"> • The <code>deviceitem</code> could not be updated

Program code

```
DeviceItem deviceItem = ...;
var descriptionUpdater = deviceItem.GetService<ModuleDescriptionUpdater>();
if (descriptionUpdater != null)
{
    if (descriptionUpdater.CanUpdate) //e.g. is update module version possible
    {
        bool result = descriptionUpdater.UpdateModuleDescription();
        .
        .
    }
}
```

7.19.21 Managing certificate**Requirement**

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal
- A project is open
See Opening a project

Application

You can use the TIA Portal Openness to handle certificates such as create and delete certificate, export and import certificate, assign and unassign certificates, and get certificate ID.

Local certificate manager

For device certificates the service LocalCertificateManager at the DeviceItem PLC is provided. The LocalCertificateManager has a local store for the certificates named LocalCertificateStore. There the certificates for the specific PLC are stored. To get the device item that owns the instance of the service, the property OwnedBy can be used.

```
DeviceItem Plc= ...;
// Get local certificate manager
var localCertificateManager = Plc.GetService<LocalCertificateManager>();
// Disable global certificate manager
localCertificateManager.EnableGlobalCertificatesStore = false;
// Get local certificate store
var localCertificateStore = localCertificateManager.LocalCertificateStore;
```

Certificate handling

Modify the following program code to create a certificate:

```
/ Create templates
var templateTls = localCertificateStore.GetCertificateTemplate(CertificateUsage.Tls);
var templateWebserver =
localCertificateStore.GetCertificateTemplate(CertificateUsage.WebServer);
var templateOpcUaServer =
localCertificateStore.GetCertificateTemplate(CertificateUsage.OpcUaServer); //
...
//Template handling and configuration is handled later
// Create certificates
var certificateTls = localCertificateStore.Certificates.Create(templateTls);
var certificateWeb = localCertificateStore.Certificates.Create(templateWebserver);
var certificateOpcUa = localCertificateStore.Certificates.Create(templateOpcUaServer);
```

Modify the following program code to delete and export a certificate:

```
var exportPath = ...;
// renew a certificate
certificateTls.Delete();
certificateTls = localCertificateStore.Certificates.Create(templateTlsNew);
certificateWeb.Export(new FileInfo(exportPath), CertificateExportFormat.Cer);
```

Modify the following program code to assign certificate to web server and OPC UA server using dynamic attributes:

```
DeviceItem opcUaSubmodule = ...;
//Find assigned certificates
var foundWebCertificate = Plc.GetAttribute("WebserverCertificate");
var foundOpcUaCertificate = opcUaSubmodule.GetAttribute("OpcUaServerCertificate");
//Assign certificates
opcUaSubmodule.SetAttribute("WebserverCertificate", certificateWeb);
opcUaSubmodule.SetAttribute("OpcUaServerCertificate", certificateOpcUa);
//Unassign certificates
opcUaSubmodule.SetAttribute("WebserverCertificate", null);
opcUaSubmodule.SetAttribute("OpcUaServerCertificate", null);
```

Modify the following program code to import certificates to the local certificate store for a PLC:

```
var certificateWithoutPwd = ...;
var certificateWithPwd = ...;
var password = new SecureString();
// ...
...
// Import certificates
// Without password
var importedCertificate1 = certificates.Import(new FileInfo(certificateWithoutPwd));
//With password
var importedCertificate2 = certificates.Import(new FileInfo(certificateWithPwd), password);
```

Modify the following program code to get certificate ID:

```
var certificateId = importedCertificate2.Id;
```

Modify the following program code to indicate if a certificate has a private key the boolean property HasPrivateKey at the certificate object can be used.

```
if(importedCertificate2.HasPrivateKey)
{
//Do something
}
```

Certificate template

The templates are used as basis for creating certificates. New templates can be created at the LocalCertificateStore using the action GetCertificateTemplate(CertificateUsage). Where CertificateUsage is an enumerated with the following possible values:

- None (not supported)
- Tls
- WebServer
- OpcUaServer
- OpcUaClient
- OpcUaClientServer

```
// Create template
var certTemplate = localCertificateStore.GetCertificateTemplate(CertificateUsage.Tls);
```

In a certificate template all properties of a certificate can be set. The access is read-write:

- Signature: The used signature algorithm of type `SignatureAlgorithm` which is an enumerated with the following possible values
 - None (not supported)
 - Sha1RSA
 - Sha256RSA

```
certTemplate.Signature = SignatureAlgorithm.Sha1RSA;
```

- SubjectAlternativeNames: Is a composition which contains objects of type `SubjectAlternativeName` (`IList` of type `<SubjectAlternativeName>`). With the action `Create(SubjectAlternativeNameType, string)` a new SAN can be created. The handed over parameters are the Type (of type `SubjectAlternativeNameType`) and the Value (of type `string`).
- SubjectAlternativeNameType
 - None (not supported)
 - Dns
 - Email
 - IP
 - Uri

```
var san1 = certTemplate.SubjectAlternativeNames.Create(SubjectAlternativeNameType.Dns,
"127.0.0.1");
var san2 = certTemplate.SubjectAlternativeNames.Create(SubjectAlternativeNameType.IP,
"192.168.178.1");
var san3 = certTemplate.SubjectAlternativeNames.Create(SubjectAlternativeNameType.Email,
"test@prov.com");
var san4 = certTemplate.SubjectAlternativeNames.Create(SubjectAlternativeNameType.Uri,
"something");
san3.Delete();
```

- SubjectCommonName: string

```
string subjectCommonName = certTemplate.SubjectCommonName;
certTemplate.SubjectCommonName = "exampleSubjectCommonName";
```

Usage of type CertificateUsage

```
var usage = certTemplate.Usage;
certTemplate.Usage = CertificateUsage.OpcUaClientServer;
```

ValidFrom: Type `DateTime`

ValidUntil: Type DateTime

```
var validFrom = certTemplate.ValidFrom;
var validUntilDateTime = new DateTime(2080, 10, 10);
certTemplate.ValidUntil = validUntilDateTime;
```

7.19.22 Blocks

7.19.22.1 Querying the "Program blocks" group

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- A PLC is determined in the project.

Program code

Modify the following program code to query the group "Program blocks":

```
private static void GetBlockGroupOfPLC(PlcSoftware plcsoftware)
//Retrieves the system group of a block
{
    PlcBlockSystemGroup blockGroup = plcsoftware.BlockGroup;
}
```

7.19.22.2 Querying the system group for system blocks

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Program code:

Modify the following program code to determine the group created for system blocks by the system:

```
PlcSoftware plcSoftware = ...
foreach (PlcSystemBlockGroup systemGroup in plcSoftware.BlockGroup.SystemBlockGroups)
{
    foreach (PlcSystemBlockGroup group in systemGroup.Groups)
    {
        PlcBlockComposition pbComposition = group.Blocks;
        foreach (PlcBlock block in pbComposition)
        {
            //add your code here
        }
    }
}
```

7.19.22.3 Enumerating system subgroups**Requirement**

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Program code: Enumerating all system subgroups

Modify the following program code to enumerate the system subgroups of all system blocks:

```
//Retrieves the system generated group for system blocks
private static void GetSystemgroupForSystemblocks(PlcSoftware plcSoftware)
{
    PlcSystemBlockGroupComposition systemBlockGroups =
plcSoftware.BlockGroup.SystemBlockGroups;
    if (systemBlockGroups.Count != 0)
    {
        PlcSystemBlockGroup sbSystemGroup = systemBlockGroups[0];
        foreach (PlcSystemBlockGroup group in sbSystemGroup.Groups)
        {
            EnumerateSystemBlockGroups(group);
        }
    }
}
private static void EnumerateSystemBlockGroups(PlcSystemBlockGroup systemBlockGroup)
{
    foreach (PlcSystemBlockGroup group in systemBlockGroup.Groups)
    {
        // recursion EnumerateSystemBlockGroups(group);
    }
}
```

Program code: Accessing a specific subgroup

Modify the following program code to access a specific subgroup:

```
private static void AccessSbGroup(PlcSystemBlockGroup systemBlockGroup)
{
    PlcSystemBlockGroup group1 = systemBlockGroup.Groups.Find("User group XYZ");
    PlcSystemBlockGroup group2 = group1.Groups.Find("User group ZYX");
}
```

See also

[Adding an external file \(Page 513\)](#)

7.19.22.4 Enumerating user-defined block groups**Requirement**

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)
- A PLC is determined in the project.

Application

Subgroups are taken into account recursively for enumeration.

Program code: Enumerating all groups

Modify the following program code to enumerate the user-defined block groups:

```
//Enumerates all block user groups including sub groups
private static void EnumerateAllBlockGroupsAndSubgroups(PlcSoftware plcsoftware)
{
    foreach (PlcBlockUserGroup blockUserGroup in plcsoftware.BlockGroup.Groups)
    {
        EnumerateBlockUserGroups(blockUserGroup);
    }
}

private static void EnumerateBlockUserGroups(PlcBlockUserGroup blockUserGroup)
{
    foreach (PlcBlockUserGroup subBlockUserGroup in blockUserGroup.Groups)
    {
        EnumerateBlockUserGroups(subBlockUserGroup);
        // recursion
    }
}
```

Program code: Accessing a group

Modify the following program code to access a selected user-defined block group:

```
//Gives individual access to a specific block user group
private static void AccessBlockusergroup(PlcSoftware plcsoftware)
{
    PlcBlockUserGroupComposition userGroupComposition = plcsoftware.BlockGroup.Groups;
    PlcBlockUserGroup plcBlockUserGroup = userGroupComposition.Find("MyUserfolder");
}
```

7.19.22.5 Enumerating all blocks

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- A PLC is determined in the project.

Application

Targeted access to a program block is possible if its name is known.

Program code: Enumerating all blocks

Modify the following program code to enumerate the blocks of all block groups:

```
private static void EnumerateAllBlocks(PlcSoftware plcsoftware)
//Enumerates all blocks
{
    foreach (PlcBlock block in plcsoftware.BlockGroup.Blocks)
    {
        // Do something...
    }
}
```

Program code: Accessing a specific block

Modify the following program code to access a specific block:

```
private static void AccessASingleBlock(PlcSoftware plcsoftware)
//Gives individual access to a block
{
    // The parameter specifies the name of the block
    PlcBlock block = plcsoftware.BlockGroup.Blocks.Find("MyBlock");
}
```

7.19.22.6 Querying information of a block/user data type

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The TIA Portal Openness API supports the querying of the following information for program and data blocks and for user data types:

- Time stamp in UTC time format.
You check the following with the time stamp:
 - When the block was last compiled.
 - When the block was last changed.
- "Consistency" attribute
The "Consistency" attribute is set to "True" in the following cases:
 - The block has been successfully compiled.
 - The block has not been changed since compilation.
 - No changes that would require re-compilation have been made to external objects.
- Programming language used (program and data blocks only)
- Block number
- Block name
- Block author
- Block family
- Block title
- Block version

See also AUTOHOTSPOT for further information.

Program code

Modify the following program code to query the information listed above:

```
private static void GetPlcBlockInformation(PlcSoftware plcSoftware)
{
    PlcBlock plcBlock = plcSoftware.BlockGroup.Blocks.Find("MyBlock");
    // Read information
    DateTime compileDate = plcBlock.CompileDate;
    DateTime modifiedDate = plcBlock.ModifiedDate;
    bool isConsistent = plcBlock.IsConsistent;
    int blockNumber = plcBlock.Number;
    string blockName = plcBlock.Name;
    ProgrammingLanguage programmingLanguage = plcBlock.ProgrammingLanguage;
    string blockAuthor = plcBlock.HeaderAuthor;
    string blockFamily = plcBlock.HeaderFamily;
    string blockTitle = plcBlock.HeaderName;
    System.Version blockVersion = plcBlock.HeaderVersion;
}
```

See also

[Importing configuration data \(Page 753\)](#)

7.19.22.7 Setting and removing protections from a block

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)
- PLC is not online.

Application

You can set or remove the password protection of a block via the PlcBlockProtectionProvider class and the PlcBlockProtectionProvider service. The service PlcBlockProtectionProvider is accessible on blocks which fulfill the following conditions:

- block is know-how protectable
- block is a code block or a global DB
- block is supported or editable in the current PLC
- block is not in readonly context
- block is not know-how protected
- block is not online
- block is not a CPU-DB
- block is not of classic encryption language, ProDiag or ProDiag-OB
- block is not an encrypted imported classic block

In case the block doesn't fulfill all conditions a null reference is returned by GetService() method.

Program code: Performing know-how protection related operations

Modify the following program code:

```
PlcBlock block = ...;
PlcBlockProtectionProvider protectionProvider =
block.GetService<PlcBlockProtectionProvider>();
if (protectionProvider != null)
{
    ... // perform know-how protection related operations here
}
```

Protect a block

Use the Protect() method to set the password to protect the porgramming block with.

```
void Protect(SecureString password)
```

Errors will occur in case

- of an attempt to protect an already protected block: An EngineeringTargetInvocationException will be thrown with the message "You can't protect an already protected object".
- of an attempt to protect with an empty string as password: An EngineeringTargetInvocationException will be thrown with the message "Password was not specified".
- of an attempt to protect a failsafe block when the failsafe-program is password protected: An EngineeringTargetInvocationException will be thrown.
- of an attempt to protect a failsafe block when the block is not called: An EngineeringTargetInvocationException will be thrown.

Unprotect a block

Use the Unprotect() method to remove the password the porgramming block is protected with.

```
void Unprotect(SecureString password)
```

Errors will occur in case

- of an attempt to unprotect an already unprotected block: An EngineeringTargetInvocationException will be thrown with the message "You can't unprotect an object without protection".
- of an attempt to unprotect with wrong password: An EngineeringTargetInvocationException will be thrown with the message "The used password was refused".
- of an attempt to protect with an empty string as password: An EngineeringTargetInvocationException will be thrown with the message "Password was not specified".

Check for invalid characters

Because you can use any characters including backspace, tab etc. to protect a block with the Protect() method, it could be impossible to remove the protection within the TIA Portal. Since the passwords are submitted as SecureString, you have to check for yourself if the provided password has illegal characters. With the GetInvalidPasswordCharacters() method you can retrieve a list of invalid characters.

```
SecureString CreatePasswordString(ProtectionProvider protectionProvider, IEnumerable<char>
contentCharacters)
{
    IList<char> invalidCharacters = protectionProvider.GetInvalidPasswordCharacters();
    SecureString password = new SecureString();
    foreach(char ch in contentCharacters)
    {
        if (!invalidCharacters.Contains(ch))
        {
            password.AppendChar(ch);
        }
        else
        {
            // at least one of the content characters is not valid
            // signal an error - e.g. throw an exception
            ...
        }
    }
    return password;
}
```

Errors will occur in case

- of an attempt to unprotect an already unprotected block: An EngineeringTargetInvocationException will be thrown with the message "You can't unprotect an object without protection".
- of an attempt to unprotect with wrong password: An EngineeringTargetInvocationException will be thrown with the message "The used password was refused".
- of an attempt to protect with an empty string as password: An EngineeringTargetInvocationException will be thrown with the message "Password was not specified".

7.19.22.8 Deleting block

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- PLC is not online.

Program code

Modify the following program code to delete a block:

```
//Runs through block group and deletes blocks
private static void DeleteBlocks(PlcSoftware plcsoftware)
{
    PlcBlockSystemGroup group = plcsoftware.BlockGroup;
    // or BlockUserGroup group = ...;
    for (int i = group.Blocks.Count - 1; i >= 0; i--)
    {
        PlcBlock block = group.Blocks[i];
        if (block != null)
        {
            block.Delete();
        }
    }
}
```

See also

[Importing configuration data \(Page 753\)](#)

7.19.22.9 Creating group for blocks

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)

Program code

Modify the following program code to create a group for blocks:

```
private static void CreateBlockGroup(PlcSoftware plcsoftware)
//Creates a block group
{
    PlcBlockSystemGroup systemGroup = plcsoftware.BlockGroup;
    PlcBlockUserGroupComposition groupComposition = systemGroup.Groups;
    PlcBlockUserGroup myCreatedGroup = groupComposition.Create("MySubGroupName");
}
```

See also

[Importing configuration data \(Page 753\)](#)

7.19.22.10 Deleting group for blocks

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- PLC is not online.

Program code

Modify the following program code to delete a group for blocks:

```
// Deletes user groups from PlcBlockSystemGroup or PlcBlockUserGroup
private static void DeleteBlockFolder(PlcSoftware plcSoftware)
{
    PlcBlockUserGroup group = plcSoftware.BlockGroup.Groups.Find("myGroup");
    //PlcBlockSystemGroup group = plcSoftware.BlockGroup;
    PlcBlockUserGroupComposition subgroups = group.Groups;
    PlcBlockUserGroup subgroup = subgroups.Find("myUserGroup");
    if (subgroup != null)
    {
        subgroup.Delete();
    }
}
```

See also

Importing configuration data (Page 753)

7.19.22.11 Accessing attributes of all blocks

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

You can set the attributes applicable for all blocks using SetAttribute() and SetAttributes() method.

SetAttributes() can operate on all objects types where writable openness attribute is available. With one input item it behaves as SetAttribute.

Parameter checking:

- Initial checks are done before the first item is being processed by global PE infrastructure: checking of input items for identical and valid attribute's name, data type mismatch. The same attribute cannot be set twice by one SetAttributes
- Sequential check happens during running of SetAttributes. The order of the processing is the same as the order of the input list elements

SetAttributes operates in only one transaction. If any attribute of input list cannot be set, the values of attributes are remaining unchanged. SetAttributes sets attributes in the same order as they can be found in the parameter list. The parameters can be dependent or independent of each other. If one parameter depends on one of the previous parameters in the list, this parameter is checked and evaluated according to the current value of the other attribute (not to the original value at the time of calling). If the SetAttributes cannot be executed, details of the error are indicated. The error message contains the first attribute name and its value which could not be managed to set and its reason.

The following program code examples were given based on the two attributes AutoNumber and Number using SetAttribute() (Refer Exporting blocks (Page 868) for all applicable attributes of blocks).

Program code: SetAttribute

```
...
PlcBlockGroup blockFolder = YourUtilities.GetFolder();
var block = blockFolder.Blocks.Find("Block_1");
if ((bool)block.GetAttribute("AutoNumber")==true)
{
    block.SetAttribute("AutoNumber",false);
}
block.SetAttribute("Number",2);
...
```

Program code: SetAttributes

```

PlcBlock block = SelectBlock("MC-Servo");
if (block != null)
{
    IList<KeyValuePair<string, object>> list = new List<KeyValuePair<string, object>>()
    {
        new KeyValuePair<string, object>("DataExchangeMode", OBDataExchangeMode.Synchronous), new
        KeyValuePair<string, object>("SynchronousApplicationCycleTime", (float)69)
    };
    try
    {
        block.SetAttributes(list);
    }
    catch (EngineeringException e)
    {
        Console.WriteLine("Exception: " + e.Message);
    }
}
}

```

7.19.22.12 Creating a ProDiag-FB

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

Openness user can use the PLCBlock composition's create action with the following parameters to create ProDiag FB.

1. Name
2. Auto number flag
3. Number
 - If "auto number flag" is true, the given block number will be set in case it is free, otherwise a new block will be generated
 - If "auto number flag" is false, the given block number will be set to the block
4. Programming language
 - If the user invoke create action with ProDiag programming language, then a new FB will be created without IDB.
 - If user invoke create action with IDB of ProDiag, than IDB of ProDiag will be created.
 - In any other not supported case, a recoverable exception is thrown.

Program code: Creating a ProDiag-FB

```
...
PlcBlockGroup blockFolder = plc.BlockGroup;
PlcBlockComposition blockComposition = blockFolder.Blocks;
if (blockComposition != null)
{
    string fbName = "ProDiag_Block";
    bool isAutoNumber = true;
    int number = 1;
    var progLang = ProgrammingLanguage.ProDiag;
    FB block = blockComposition.CreateFB(fbName, isAutoNumber, number, progLang);
    string iDBName="ProDiag_IDB";
    string instanceOfName = fbName;
    InstanceDB iDbBlock = blockComposition.CreateInstanceDB(iDBName, isAutoNumber, number,
instanceOfName);
}
...
...
```

See also

[Accessing supervisions and properties of ProDiag-FB \(Page 511\)](#)

7.19.22.13 Accessing supervisions and properties of ProDiag-FB

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)

Accessing supervisions of User-FB

Openness user can access the supervisions at FB by using the following code snippet. Every FB has the list of supervisions including Classic and Plus PLCs.

Program code: Accessing supervisions of ProDiag-FB

```
...
PlcBlock iDB = plc.BlockGroup.Blocks.Find("FB_Block_DB");
string fbName = iDB.GetAttribute("InstanceOfName").ToString();
FB fb = (FB)plc.BlockGroup.Blocks.Find(fbName);
if (fb.Supervisions.Count > 0) Console.WriteLine ("Contains supervisions");
else
Console.WriteLine ("Does not contains supervisions");
...
...
```

Accessing the attributes of FB block

Openness user can set AssignedProDiagFB at InstanceDB via the attribute AssignedProDiagFB (Refer Exporting blocks (Page 868)). The user can use GetAttribute(), GetAttributes() and SetAttribute() method for accessing the attributes. The user cannot use SetAttributes() method for setting the attributes for more than one attribute. TIA Portal Openness throws an exception for using SetAttributes() method.

If the attribute is not supported (in the given block), recoverable user exception is thrown. If there is no assigned ProDiag-Block set, GetAttribute() returns an empty string.

Program code: Getting and setting the assigned ProDiag-FB at and IDB

```
...
PlcBlockGroup blockFolder = plc.BlockGroup;
PlcBlock instanceDB = blockFolder.Blocks.Find("IDB");
PlcBlock plcProdiag = blockFolder.Blocks.Find("block_Prodiag");
instanceDB.SetAttribute("AssignedProDiagFB", plcProdiag.Name);
var assignedProDiagFB = instanceDB.GetAttribute("AssignedProDiagFB");
...
...
```

See also

[Creating a ProDiag-FB \(Page 510\)](#)

7.19.22.14 Reading ProDiag-FB blocks and attributes

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- You have opened a project via a TIA Portal Openness application
See Opening a project (Page 106)

Application

You can use the TIA Portal Openness to read the ProDiag function block version, and other ProDiag related attribute values. You can use GetAttribute () and GetAttributes () methods to read the ProDiag FBs language specific attributes present.

Attributes

The following attributes are supported by ProDiag-FB in Openness:

Attributes	Type
ProDiagVersion	Version
InitialValueAcquisition	bool
UseCentralTimeStamp	bool

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.19.22.15 Adding an external file

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- You have opened a project via a TIA Portal Openness application:
See Opening a project (Page 106)

Application

You can add an external file to a PLC. This external file is stored in the file system under the defined path.

The following formats are supported:

- STL
- SCL
- DB
- UDT

Note

Accessing groups in the "External source files" folder is not supported.

An exception is thrown if you specify a file extension other than *.AWL, *.SCL, *.DB or *.UDT.

Program code

Modify the following program code to create an external file in the "External source files" folder from a block.

```
private static void CreateBlockFromFile(PlcSoftware plcSoftware)
// Creates a block from a AWL, SCL, DB or UDT file
{
    PlcExternalSource externalSource =
plcSoftware.ExternalSourceGroup.ExternalSources.CreateFromFile("SomeBlockNameHere", "SomePathHere");
}
```

7.19.22.16 Generate source from block

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- PLC is not online.

Application

The TIA Portal Openness API interface supports the generation of sources in UTF-8 from STL or SCL blocks, data blocks and PLCTypes (user data types). To generate a source file of a block, invoke the method `GenerateSource` on the `PlcExternalSourceSystemGroup` instance.

The scale of the generated source file depends on the generation option of this function:

- `GenerateOptions.None`
Generate source from provided blocks only.
- `GenerateOptions.WithDependencies`
Generate source including all dependent objects.

The interface `Siemens.Engineering.SW.ExternalSources.IGenerateSource` indicates that a source can be generated.

Only the STL and SCL programming languages are supported for blocks. Exceptions are thrown in the following cases:

- Programming language is not STL or SCL
- A file of the same name already exists at the target location

Only the `".udt"` file extension is supported for user data types. Exceptions are thrown in the following cases:

- The file extension is not `".db"` for DBs
- The file extension is not `".awl"` for STL blocks
- The file extension is not `".scl"` for SCL blocks

Program code

Modify the following program code to generate source files from blocks and types:

```
//method declaration
...
PlcExternalSourceSystemGroup.GenerateSource

(IEnumerable<Siemens.Engineering.SW.ExternalSources.IGenerateSource>
plcBlocks, FileInfo sourceFile, GenerateOptions generateOptions);
...
//examples
...
var blocks = new List<PlcBlock>(){block1};
var fileInfo = new FileInfo(@"C:\temp\SomePathHere.scl");

PlcExternalSourceSystemGroup systemGroup = ...;

systemGroup.GenerateSource(blocks, fileInfo, GenerateOptions.WithDependencies);

// exports all blocks and with all their dependencies(e.g. called blocks, used DBs or UDTs)
// as ASCII text into the provided source file.
...
or
..
var types = new List<PlcType>(){udt1};
var fileInfo = new FileInfo(@"C:\temp\SomePathHere.udt");

PlcExternalSourceSystemGroup systemGroup = ...;

systemGroup.GenerateSource(types, fileInfo, GenerateOptions.WithDependencies );

// exports all data types and their used data types into the provided source file.
...
```

See also

[Importing configuration data \(Page 753\)](#)

7.19.22.17 Generating blocks from source

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)
- PLC is not online.

Application

You can generate blocks from all external files in the "External source files" group. Only external files with the format ASCII are supported.

Note

Access to groups in the "External source files" folder is not supported.

Existing blocks are overwritten.

An Exception is thrown if an error occurs during the calling. The first 256 characters of each error message are contained in the notification of the Exception. The project is reset to the processing state prior to the execution of the `GenerateBlocksFromSource` method.

Program code

Modify the following program code to generate blocks from all external files in the "External source files" group.

```
// Creates a block from an external source file
PlcSoftware plcSoftware = ...;
foreach (PlcExternalSource plcExternalSource in
plcSoftware.ExternalSourceGroup.ExternalSources)
{
    plcExternalSource.GenerateBlocksFromSource();
}
```

7.19.22.18 Generating from source of known source format

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- PLC is not online

Application

The generator is basically able to create an object (block/UDT) but 'something inside' makes that the generated block/UDT won't be compilable.

Typical missing or invalid parts what makes the block / type not compilable:

- reference to a not existing type
- missing symbol used

In case of generation errors the current implementation of 'GenerateBlocksFromSource' void GenerateBlocksFromSource():

- throws a recoverable exception (due to failed generating)
- Includes the error message(s) of the generating to the exception's message
- and deletes the generated block/UDT

Due to the Openness Long Term Stability a new overload of GenerateBlocksFromSource is available:

`IList<IEngineeringObject> GenerateBlocksFromSource(GenerateBlockOptions option)`

If this new 'GenerateBlocksFromSource' called with `GenerateBlockOptions.KeepOnError`, it

- doesn't throw recoverable exception (regardless the generating results)
- doesn't provide information about the generation result of any generated block(s)
- returns with an `IList` of `IEngineeringObjects` of the generated blocks (with or without generation errors)

If this new 'GenerateBlocksFromSource' called with `GenerateBlockOptions.None` and generation wasn't flawless, it

- throws a recoverable exception (due to failed generating)
- includes the error message(s) of the generating to the exception's message
- and deletes the generated block/UDT
 - this is just the same behavior as the current implementation does

In case of flawless generation, it

- doesn't throw any exception
- doesn't provide information about the generation result of any generated blocks
- returns with an `IList` of `IEngineeringObjects` of the generated blocks
 - this is the same behavior as having called it with `GenerateBlockOptions.KeepOnError`.

Program code

```
try
{
    IList<IEngineeringObject> generatedObjects =
externalSource.GenerateBlocksFromSource(GenerateBlockOptions.KeepOnError);
    foreach (IEngineeringObject engineeringObject in generatedObjects)
{
    CompilerResult compilerResult = null;
    string objectName = null;
    if (engineeringObject is PlcBlock)
    {
        // handle case for PlcBlock
        // e.g. retrieve the compiler result
        PlcBlock block = (PlcBlock)engineeringObject;
        objectName = block.Name;
        compilerResult = block.Compile();
    }
    else if (engineeringObject is PlcType)
    {
        // handle case for PlcType
        // e.g. retrieve the compiler result
        PlcType plcType = (PlcType)engineeringObject;
        objectName = plcType.Name;
        compilerResult = plcType.Compile();
    }
    // handle the compiler result
    if (compilerResult != null)
    {
        if (compilerResult.State == CompilerResultState.Error)
        {
            Console.WriteLine("Object '{0}' could not be compiled successfully!", objectName);
            Console.WriteLine("Number of compiler errors: {0}", compilerResult.ErrorCount);
            foreach (CompilerResultMessage compilerResultMessage in compilerResult.Messages)
            {
                Console.WriteLine(compilerResultMessage.Description);
            }
        }
        else
        {
            Console.WriteLine("Object '{0}' could be compiled successfully.", objectName);
        }
    }
}
catch (RecoverableException exception)
{
    // handle recoverable exception
    Console.WriteLine(exception.Message);
}
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.19.22.19 Deleting user data type

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)
- PLC is not online.

Program code

Modify the following program code to delete a user type:

```
private static void DeleteUserDataType(PlcSoftware plcSoftware)
{
    PlcTypeSystemGroup typeGroup = plcSoftware.TypeGroup;
    PlcTypeComposition dataTypes = typeGroup.Types;
    PlcType dataType = dataTypes.Find("DataTypeName");
    if (dataType != null)
    {
        dataType.Delete();
    }
}
```

See also

[Importing configuration data \(Page 753\)](#)

7.19.22.20 Deleting an external file

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
see [Connecting to the TIA Portal \(Page 76\)](#)
- You have opened a project via a TIA Portal Openness application:
see [Opening a project \(Page 106\)](#)
- PLC is not online

Program code

Modify the following program code to delete an external file in the "External source files" group.

Note

Access to groups in the "External source files" group is not supported.

```
// Deletes an external source file
private static void DeleteExternalSource(PlcSoftware plcSoftware)
{
    PlcExternalSource externalSource =
plcSoftware.ExternalSourceGroup.ExternalSources.Find("myExternalsource");
    externalSource.Delete();
}
```

7.19.22.21 Starting the block editor

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- Instance of the TIA Portal is opened with user interface.

Program code

Modify the following program code to start the associated editor for an object reference of the type `PlcBlock` in the TIA Portal instance:

```
//Opens a block in a block editor
private static void StartBlockEditor(PlcSoftware plcSoftware)
{
    PlcBlock plcBlock = plcSoftware.BlockGroup.Blocks.Find("MyBlock");
    plcBlock.ShowInEditor();
}
```

7.19 Functions for accessing the data of a PLC device

Modify the following program code to open the associated editor for an object reference of the type `PlcType` in the TIA Portal instance:

```
//Opens a udt in udt editor
private static void StartPlcTypEditor(PlcSoftware plcSoftware)
{
    PlcTypeComposition types = plcSoftware.TypeGroup.Types;
    PlcType udt = types.Find("my_udt");
    udt.ShowInEditor();
}
```

See also

[Importing configuration data \(Page 753\)](#)

7.19.22.22 Changing blocks using fingerprints

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- You have opened a project with your TIA Portal Openness application.
See [Opening a project \(Page 106\)](#)

Application

You can use the TIA Portal Openness to detect changes inside of blocks or UDT. You can achieve this by comparing the fingerprints of the object. A fingerprint instance contains an `FingerprintId` which defines the kind of fingerprint and the fingerprint value as a string. All provided fingerprints consider only user input, no compilation result, or any other change made by the system.

The enumeration `FingerprintId` lists all kind of fingerprints supported in Openness:

Value	Description
Code	Considers all changes in the code inside the body of the block. It does not consider the compilation result.
Interface	Considers all changes in the interface of a block. Including start values of a DB
Properties	Considers changes in the properties of a block. e.g. name, number
Comments	Considers changes in the comments of a block. In case of OBs the fingerprint also changes when the list of available languages in project language setting changes
LibraryType	Exists when a block is connected to a library type

Value	Description
Texts	With V15 SP1 this fingerprint only exists for Graph blocks
Alarms	Exists when a block uses alarming.
Supervision	Exists when a block contains supervision
TechnologyObject	Exists only for technology object DBs
Events	Exists only for OB
TextualInterface	Exists when the block has a textual interface

You need to use the FingerprintProvider service to retrieve the fingerprints of an object. It is available for blocks (FB, FC, OB, DB), and UDTs, but not for tag tables. The FingerprintProvider calculates and returns all available fingerprints of an object with each GetFingerprints call. In order to ensure correct fingerprints, the block or UDT needs to be consistent before calling fingerprints. Otherwise, an RecoverableException is thrown. When a fingerprint is still invalid after its calculation, an RecoverableException is thrown.

Program code

Modify the following program code to get the fingerprint instance:

```
PlcBlock block = ...;
FingerprintProvider provider = block.GetService<FingerprintProvider>();
IList<Fingerprint> fingerprints = provider.GetFingerprints();
foreach(var fingerprint in fingerprints)
{
    string fpValue = fingerprint.Value;
    FingerprintId fpId = fingerprint.Id;
}
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.19.22.23 Generating/deleting blocks for user defined pages

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open
See [Opening a Project \(Page 106\)](#)

Application

You can use the TIA Portal Openness to create user defined web pages within the web server of PLCs. The data for these user defined pages are stored in specific system generated data blocks of the PLC. You can also edit and delete these blocks manually. But if you modify any data or properties in these data blocks the web server will not function correctly.

In TIA Portal Openness, the new service WebserverUserDefinedPages is added at the DeviceItem PLC. You can use this service to generate the DBs for user defined pages: List<PLCBlock> GenerateBlocks(Arguments). When this function is called the corresponding DBs are generated and a list of generated DBs is returned to user.

There are two possible overloads for this function:

- List<PLCBlock> WebserverUserDefinedPages.GenerateBlocks(WebDBGenerateOptions GenerateOptions);
- List<PLCBlock> WebserverUserDefinedPages.GenerateBlocks(System.IO.DirectoryInfo htmlDirectory, System.IO.FileInfo defaultHTMLPage, string applicationName, WebDBGenerateOptionsGenerateOptions);

The parameter values for HTML directory, HTML page and application name are not stored in the project data. So the values of the corresponding dynamic attributes are not changed.

Parameter	Data type	Type	Descriptions
GenerateOptions	WebDBGenerateOptions (enumeration)	Mandatory	Possible values: <ul style="list-style-type: none"> • None - If DBs for user defined pages already have been created before, no action is performed and an exception is thrown • Override - If DBs for user defined pages already have been created before, they are deleted and the new DBs are generated
htmlDirectory	System.IO.DirectoryInfo	Optional	Specifies the HTML directory independently from the value of the dynamic attribute WebserverHTMLDirectory
defaultHTMLPage	System.IO.FileInfo	Optional	Specifies the default HTML page independently from the value of the dynamic attribute WebserverDefaultHTMLPage
applicationName	string	Optional	Specifies the application name independently from the value of the dynamic attribute WebserverApplicationName

GenerateBlocks() will throw recoverable exception:

- If web server is disabled (WebserverActive == False) - "Webserver has to be enabled"
- If WebserverHTMLDirectory is empty or the path is invalid - "HTML directory has invalid path"

- If WebserverdefaultHTMLPage is empty or the path is invalid - "Default HTML page has invalid path"
- When the WebserverHTMLDirectory is too long - "HTML directory has too large data"
- The Application Name is invalid - "Application Name is invalid"
- The Dynamic Content is invalid - "Dynamic Content is invalid"
- The Control DB number is invalid - "Control DB number is invalid"
- The DB Start Number is invalid - "DB Start Number is invalid"
- Blocks are already present: If the user tries to generate a block with a name which already exists- ""Delete existing blocks before generating new blocks" (Only thrown if WebDBGenerateOptions.None is used)

There is no specific function to delete all blocks connected to the user defined pages. You have to manually delete the blocks that are handed back to them by the generate action or navigate to the corresponding DBs as it is already possible in TIA Portal Openness.

Program code

Modify the following program code to generate a block for user defined page:

```
DeviceItem deviceItem = ...;
var WebserverUserDefinedPagesService = deviceItem.GetService<WebserverUserDefinedPages>();
List<PLCBlock> blocks =
WebserverUserDefinedPagesService.GenerateBlocks(WebDBGenerateOptions.None);
List<PLCBlock> blocks = WebserverUserDefinedPagesService.GenerateBlocks(htmlDirectory,
defaultHTMLPage, applicationName, WebDBGenerateOptions.Override);
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.19.23 Technology objects

7.19.23.1 Overview of functions for technology objects

TIA Portal Openness supports a selection of technology object functions for defined tasks that you can call outside the TIA Portal by means of the Public API.

You get the code components that have to be adapted for each task.

Functions

The following functions are available for technology objects:

- Querying the composition of technology objects (Page 529)
- Creating technology object (Page 529)
- Deleting technology object (Page 530)
- Compiling technology object (Page 531)
- Enumerating technology object (Page 532)
- Finding technology object (Page 533)
- Enumerating parameters of technology object (Page 534)
- Finding parameters of technology object (Page 534)
- Reading parameters of technology object (Page 535)
- Writing parameters of technology object (Page 536)

See also

[Standard libraries \(Page 43\)](#)

[Applications \(Page 37\)](#)

7.19.23.2 Overview of technology objects and versions

Technology objects

The following table shows the available technology objects in the Public API.

CPU	FW	Technology object	Version of technology object
S7-1200	≥ V4.2	TO_PositioningAxis	V6.0
		TO_CommandTable	
		PID_Compact	V2.3
		PID_3Step	
		PID_Temp	V1.1

CPU	FW	Technology object	Version of technology object
S7-1500	< V2.0	High_Speed_Counter	V3.0
		SSI_Absolute_Encoder	V2.0
	≥ V2.0	TO_SpeedAxis	≥ V3.0
		TO_PositioningAxis	
		TO_ExternalEncoder	
		TO_SynchronousAxis	
		TO_OutputCam	
		TO_CamTrack	
		TO_MeasuringInput	
		TO_Cam (S7-1500T) ¹⁾	
		TO_Kinematics (S7-1500T)	V4.0
		High_Speed_Counter	≥ V3.0
		SSI_Absolute_Encoder	≥ V2.0
		PID_Compact	≥ V2.3
		PID_3Step	V2.3
		PID_Temp	V1.1
		CONT_C	
		CONT_S	
		TCONT_CP	
		TCONT_S	
S7-300/400	Any	CONT_C	V1.1
		CONT_S	
		TCONT_CP	
		TCONT_S	
		TUN_EC ²⁾	
		TUN_ES ²⁾	
		PID_CP ²⁾	V2.0
		PID_ES ²⁾	
		AXIS_REF	

1) The technology object does not support the following Openness functions: Writing parameters.

2) The technology object does not support the following Openness functions: Enumerating parameters, Finding parameters, Reading parameters, Writing parameters.

Note

S7-1500 Motion Control

The technology objects TO_OutputCam, TO_CamTrack and TO_MeasuringInput on S7-1500 are handled separately.

You can find further information in section "S7-1500 Motion Control (Page 545)".

7.19.23.3 Overview of data types

The data types of technology object parameters in TIA Portal are mapped to C# data types in the Public API.

Data types

The following table shows the data type mapping:

Format	Data type in TIA Portal	Data type in C#
Binary numbers	Bool	bool
	BBool	bool
	Byte	byte
	Word	ushort
	DWord	uint
	LWord	ulong
	SInt	sbyte
	Int	short
	DInt	int
	LInt	long
Integers	USInt	byte
	UInt	ushort
	UDInt	uint
	ULInt	ulong
	Floating-point numbers	float
	LReal	double
	Time	double
	Character strings	char
	WChar	char
	String	string
Hardware data types	WString	string
	HW_*	ushort
Hardware data types	Block_*	ushort

* Placeholder for device type extension in TIA Portal project

7.19.23.4 Querying the composition of technology objects

Requirement

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- A PLC is determined in the project.
See Querying PLC and HMI targets (Page 178)

Program code

Modify the following program code to get all technology objects of a PLC:

```
// Retrieves all technology objects of a PLC
private static void GetTechnologicalObjectsOfPLC(PlcSoftware plcSoftware)
{
    TechnologicalInstanceDBGroup technologicalObjectGroup =
plcSoftware.TechnologicalObjectGroup;
    TechnologicalInstanceDBCComposition technologicalObjects =
technologicalObjectGroup.TechnologicalObjects;
}
```

See also

Standard libraries (Page 43)

7.19.23.5 Creating technology object

Requirement

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- A PLC is determined in the project.
See Querying PLC and HMI targets (Page 178)

Application

Only technology objects that are listed in the section Overview of technology objects and versions (Page 526) can be created. An exception is thrown for unsupported technology objects or invalid parameters.

Note**S7-1500 Motion Control**

The technology objects TO_OutputCam, TO_CamTrack and TO_MeasuringInput on S7-1500 are handled separately.

You can find further information in section "S7-1500 Motion Control (Page 545)".

Program code

Modify the following program code to create a technology object and add it to an existing PLC:

```
// Create a technology object and add to technology object composition
private static void CreateTechnologicalObject(PlcSoftware plcSoftware)
{
    TechnologicalInstanceDBComposition technologicalObjects =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects;

    string nameOfTO = "PID_Compact_1"; // How the technology object should be named
    string typeOfTO = "PID_Compact"; // How the technology object type is called, e.g. in
    // "Add new technology object"-dialog
    Version versionOfTO = new Version("2.3"); // Version of technology object
    TechnologicalInstanceDB technologicalObject = technologicalObjects.Create(nameOfTO,
typeOfTO, versionOfTO);
}
```

Possible values and combinations of name, type and version of the technology object can be found in the section Overview of technology objects and versions (Page 526).

See also

[Standard libraries \(Page 43\)](#)

7.19.23.6 Deleting technology object**Requirement**

- The Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)
- A PLC is determined in the project.
See [Querying PLC and HMI targets \(Page 178\)](#)
- The technology object exists.
See [Finding technology object \(Page 533\)](#)

Program code

Modify the following program code to delete a technology object:

```
// Delete a technology object from DB composition and from PLC
private static void DeleteTechnologicalObject(TechnologicalInstanceDB technologicalObject)
{
    technologicalObject.Delete();
}
```

See also

[Standard libraries \(Page 43\)](#)

7.19.23.7 Compiling technology object

Requirement

- The Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)
- A PLC is determined in the project.
See [Querying PLC and HMI targets \(Page 178\)](#)
- The technology object exists.
See [Creating technology object \(Page 529\)](#)

Program code: Compiling a technology object

Modify the following program code to compile a technology object:

```
// Compile a single technology object
private static void CompileSingleTechnologicalObject(TechnologicalInstanceDB
technologicalObject)
{
    ICompilable singleCompile = technologicalObject.GetService<ICompilable>();
    CompilerResult compileResult = singleCompile.Compile();
}
```

Program code: Compiling the technology object group

Modify the following program code to compile the technology object group:

```
// Compile technology object group
private static void CompileTechnologicalObjectGroup(PlcSoftware plcSoftware)
{
    TechnologicalInstanceDBGroup technologicalObjectGroup =
plcSoftware.TechnologicalObjectGroup;
    ICompilable groupCompile = technologicalObjectGroup.GetService<ICompilable>();
    CompilerResult compileResult = groupCompile.Compile();
}
```

Compile results

Technology objects compilation results are stored recursively.

You can find an example of recursive evaluation of compilation results in the section "Compiling a project (Page 129)".

See also

[Standard libraries \(Page 43\)](#)

7.19.23.8 Enumerating technology object

Requirement

- The Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)
- A PLC is determined in the project.
See [Querying PLC and HMI targets \(Page 178\)](#)

Program code

Modify the following program code to enumerate technology objects:

```
// Enumerate all technology objects
private static void EnumerateTechnologicalObjects(PlcSoftware plcSoftware)
{
    TechnologicalInstanceDBComposition technologicalObjects =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects;
    foreach (TechnologicalInstanceDB technologicalObject in technologicalObjects)
    {
        // Do something ...
    }
}
```

See also

[Standard libraries \(Page 43\)](#)

7.19.23.9 Finding technology object

Requirement

- The Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)
- A PLC is determined in the project.
See [Querying PLC and HMI targets \(Page 178\)](#)

Program code

Modify the following program code to find a specific technology object:

```
// Find a specific technology object by its name
private static void FindTechnologicalObject(PlcSoftware plcSoftware)
{
    TechnologicalInstanceDBComposition technologicalObjects =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects;
    string nameOfTO = "PID_Compact_1";
    TechnologicalInstanceDB technologicalObject = technologicalObjects.Find(nameOfTO);
}
```

See also

[Standard libraries \(Page 43\)](#)

7.19.23.10 Enumerating parameters of technology object

Requirement

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- A PLC is determined in the project.
See Querying PLC and HMI targets (Page 178)
- A technology object exists.
See Creating technology object (Page 529) or Finding parameters of technology object (Page 534)
- The technology object (Page 526) supports this function.

Program code

Modify the following program code to enumerate parameters of a specific technology object:

```
// Enumerate parameters of a technology object
private static void EnumerateParameters(PlcSoftware plcSoftware)
{
    string nameOfTO = "PID_Compact_1";
    TechnologicalInstanceDB technologicalObject =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects.Find(nameOfTO);

    foreach (TechnologicalParameter parameter in technologicalObject.Parameters)
    {
        // Do something ...
    }
}
```

See also

[Standard libraries \(Page 43\)](#)

[Finding technology object \(Page 533\)](#)

7.19.23.11 Finding parameters of technology object

Requirement

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

- A PLC is determined in the project.
See Querying PLC and HMI targets (Page 178)
- A technology object exists.
See Creating technology object (Page 529)
- The technology object (Page 526) supports this function.

Program code

Modify the following program code to find parameters of a specific technology object:

```
// Find parameters of a technology object
private static void FindParameterOfTechnologicalObject(PlcSoftware plcSoftware)
{
    string nameOfTO = "PID_Compact_1";
    TechnologicalInstanceDB technologicalObject =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects.Find(nameOfTO);

    string nameOfParameter = "Config.InputUpperLimit";
    TechnologicalParameter parameter =
technologicalObject.Parameters.Find(nameOfParameter);
}
```

Parameters of different technology objects

- Parameters of S7-1200 Motion Control (Page 537)
- Parameters of S7-1500 Motion Control (Page 545)
- Parameters of PID Control (Page 563)
- Parameters of Counting (Page 564)
- Parameters of Easy Motion Control (Page 564)

See also

- Standard libraries (Page 43)
- Finding technology object (Page 533)

7.19.23.12 Reading parameters of technology object

Requirement

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

7.19 Functions for accessing the data of a PLC device

- A PLC is determined in the project.
See Querying PLC and HMI targets (Page 178)
- A technology object exists.
See Creating technology object (Page 529)
- The technology object (Page 526) supports this function.

Program code

Modify the following program code to read parameters of a specific technology object:

```
// Read parameters of a technology object
private static void ReadParameterOfTechnologicalObject(PlcSoftware plcSoftware)
{
    string nameOfTO = "PID_Compact_1";
    TechnologicalInstanceDB technologicalObject =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects.Find(nameOfTO);

    string nameOfParameter = "Config.InputUpperLimit";
    TechnologicalParameter parameter =
technologicalObject.Parameters.Find(nameOfParameter);

    // Read from parameter
    string name = parameter.Name;
    object value = parameter.Value;
}
```

See also

[Standard libraries \(Page 43\)](#)

[Finding technology object \(Page 533\)](#)

7.19.23.13 Writing parameters of technology object**Requirement**

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- A PLC is determined in the project.
See Querying PLC and HMI targets (Page 178)
- A technology object exists.
See Creating technology object (Page 529)
- The technology object (Page 526) supports this function.

Exception

An EngineeringException is thrown if:

- You set a new value for a parameter that does not provide write access.
- A new value for a parameter is of an unsupported type.

Program code

Modify the following program code to write parameters of a specific technology object:

```
// Write parameters of a technology object
private static void WriteParameterOfTechnologicalObject(PlcSoftware plcSoftware)
{
    string nameOfTO = "PID_Compact_1";
    TechnologicalInstanceDB technologicalObject =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects.Find(nameOfTO);

    string nameOfParameter = "Config.InputUpperLimit";
    TechnologicalParameter parameter =
technologicalObject.Parameters.Find(nameOfParameter);

    // Write to parameter if the value is writable
    object value = 3.0;
    parameter.Value = value;
}
```

Parameters of different technology objects

[Parameters of S7-1200 Motion Control \(Page 537\)](#)

[Parameters of S7-1500 Motion Control \(Page 545\)](#)

[Parameters of PID Control \(Page 563\)](#)

[Parameters of Counting \(Page 564\)](#)

[Parameters of Easy Motion Control \(Page 564\)](#)

See also

[Standard libraries \(Page 43\)](#)

[Finding technology object \(Page 533\)](#)

7.19.23.14 S7-1200 Motion Control

Changing version of openness engineering library

If you use "Openness\PublicAPI\V14 SP1\Siemens.Engineering.dll" with TIA Portal V15, your existing openness application will still work.

7.19 Functions for accessing the data of a PLC device

If you change to "Openness\PublicAPI\V15\Siemens.Engineering.dll" with TIA Portal V15, you have to adapt all accesses to array tags for S7-1200 Motion Control.

The affected arrays for TO_PositioningAxis are listed in the following table:

Access in Openness < V15	Access in Openness ≥ V15
_Sensor.Sensor[1].<all tags>	_Sensor[1].<all tags>
ControlPanel.Input.Command.Command[1].<all tags>	ControlPanel.Input.Command[1].<all tags>
ControlPanel.Output.Command.Command[1].<all tags>	ControlPanel.Output.Command[1].<all tags>
Internal.Internal[n].<all tags>	Internal[n].<all tags>
Sensor.Sensor[1].<all tags>	Sensor[1].<all tags>
StatusSensor.StatusSensor[1].<all tags>	StatusSensor[1].<all tags>

The affected arrays for TO_CommandTable are listed in the following table:

Access in Openness < V15	Access in Openness ≥ V15
Command.Command[n].<all tags>	Command[n].<all tags>

Connecting PROFIdrives by hardware address

Requirement

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76).
- A project is open.
See Opening a project (Page 106).
- A S7-1200 PLC is determined in the project.
- A PROFIdrive is available in the project and connected with the S7-1200 PLC.
- The technology object exists.
See Creating technology object (Page 529).

Program code

Modify the following program code to connect a PROFIdrive by hardware address to the "TO_PositioningAxis".

```
//An instance of the technology object axis is already available in the program before
private static void ConnectingDrive(TechnologicalInstanceDB technologicalObject)
{
    //Set axis to PROFIdrive mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 1;

    //Set axis to drive mode
    technologicalObject.Parameters.Find("_Actor.Interface.DataConnection").Value = 0;

    //Set connection to address of drive. The output will be set automatically.
    technologicalObject.Parameters.Find("_Actor.Interface.ProfiDriveIn").Value = "%I68.0";
    technologicalObject.Parameters.Find("Sensor[1].Interface.Number").Value = 1;
    // 1 = Encoder1, 2 = Encoder2;
}
```

Connecting encoders for PROFIdrives by hardware address

Requirement

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76).
- A project is open.
See Opening a project (Page 106).
- A S7-1200 PLC is determined in the project.
- A PROFIdrive is available in the project and connected with the S7-1200 PLC.
- The technology object exists.
See Creating technology object (Page 529).

Program code

Modify the following program code to connect an encoder by hardware address to the "TO_PositioningAxis":

```
//An instance of the technology object axis is already available in the program before
private static void ConnectingEncoder(TechnologicalInstanceDB technologicalObject)
{
    //Set axis to PROFIdrive mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 1;

    //Set the encoder mode
    technologicalObject.Parameters.Find("_Sensor[1].Interface.EncoderConnection").Value =
7;

    //Set axis to use PROFINET encoder
    technologicalObject.Parameters.Find("_Sensor[1].Interface.DataConnection").Value = 0;

    //Set connection to address of drive. The output will be set automatically.
    technologicalObject.Parameters.Find("_Sensor[1].Interface.ProfiDriveIn").Value =
"%I68.0";
    technologicalObject.Parameters.Find("Sensor[1].Interface.Number").Value = 1;
    // 1 = Encoder1, 2 = Encoder2;
}
```

Connecting analog drives by hardware address

Requirement

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76).
- A project is open.
See Opening a project (Page 106).
- A S7-1200 PLC is determined in the project.
- An analog drive is available in the project and connected with the S7-1200 PLC.
- The technology object exists.
See Creating technology object (Page 529).

Program code

Modify the following program code to connect an analog drive by hardware address to the "TO_PositioningAxis":

```
//An instance of the technology object axis is already available in the program before
private static void ConnectingEncoder(TechnologicalInstanceDB technologicalObject)
{
    //Set axis to analog drive mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 0;

    //Set axis to drive mode
    technologicalObject.Parameters.Find("_Actor.Interface.DataConnection").Value = 0;

    //Set connection to analog adress of drive
    technologicalObject.Parameters.Find("_Actor.Interface.Analog").Value = "%QW64";
}
```

Connecting encoders for analog drives by hardware address

Requirement

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76).
- A project is open.
See Opening a project (Page 106).
- A S7-1200 PLC is determined in the project.
- An analog drive is available in the project and connected with the S7-1200 PLC.
- The technology object exists.
See Creating technology object (Page 529).

Program code

Modify the following program code to connect an encoder by hardware address to the "TO_PositioningAxis":

```
//An instance of the technology object axis is already available in the program before
//Connecting by High Speed Counter mode
private static void ConnectingEncoder(TechnologicalInstanceDB technologicalObject)
{
    //Set axis to analog drive mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 0;

    //Set encoder for high-speed counter mode
    technologicalObject.Parameters.Find("_Sensor[1].Interface.EncoderConnection").Value =
4;
    technologicalObject.Parameters.Find("_Sensor[1].Interface.HSC.Name").Value = "HSC_1";
}

//An instance of the technology object axis is already available in the program before
//Connecting by PROFINET/PROFIBUS telegram
private static void ConnectingEncoder(TechnologicalInstanceDB
technologicalObject)
{
    //Set axis to analog drive mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 0;
    //Set encoder for PROFINET/PROFIBUS mode
    technologicalObject.Parameters.Find("_Sensor[1].Interface.EncoderConnection").Value =
7;
    technologicalObject.Parameters.Find("_Sensor[1].Interface.DataConnection").Value =
"Encoder";
    technologicalObject.Parameters.Find("_Sensor[1].Interface.ProfiDriveIn").Value =
"%I68.0";
    technologicalObject.Parameters.Find("Sensor[1].Interface.Number").Value = 1;
    // 1 = Encoder1, 2 = Encoder2;
}
```

Connecting drives by data block

Requirement

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76).
- A project is open.
See Opening a project (Page 106).
- A S7-1200 PLC is determined in the project.

- A data block is available in the project and set to "Not optimized".
For the PROFIdrive axis type, the data block contains a tag of the type e. g. PD_TEL3.
For an analog drive, the data block contains a tag with the word data type.
- The technology object exists.
See Creating technology object (Page 529).

Program code

Modify the following program code to connect a PROFIdrive by data block to the "TO_PositioningAxis".

```
//An instance of the technology object axis is already available in the program before
private static void ConfigureDrivewithDataBlock(TechnologicalInstanceDB
technologicalObject)
{
    //Set axis to PROFIdrive mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 1;

    //Set axis to data block mode
    technologicalObject.Parameters.Find("_Actor.Interface.DataConnection").Value = 1;

    //Set the tag in the data block
    technologicalObject.Parameters.Find("_Actor.Interface.DataBlock").Value =
"Data_block_1.Member_of_type_PD_TEL3";
}
```

Program code

Modify the following program code to connect an analog drive by data block to the "TO_PositioningAxis".

```
//An instance of the technology object axis is already available in the program before
//Connecting an analog drive with data block.
private static void ConfigureDrivewithDataBlock(TechnologicalInstanceDB
technologicalObject)
{
    //Set axis to analog mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 0;

    //Set the tag in the data block
    technologicalObject.Parameters.Find("_Actor.Interface.Analog").Value =
>Data_block_1.Static_1";
}
```

Connecting encoders by data block

Requirement

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76).
- A project is open.
See Opening a project (Page 106).
- A S7-1200 PLC is determined in the project.
- A data block is available in the project and set to "Not optimized".
In case of PROFIdrive the data block contains a tag of the type e. g. PD_TEL3
- The technology object exists.
See Creating technology object (Page 529).

Program code

Modify the following program code to connect an encoder by data block:

```
//An instance of the technology object axis is already available in the program before
private static void ConfigureEncoderWithDataBlock(TechnologicalInstanceDB
technologicalObject)
{
    //Set axis to PROFIdrive mode depending by axis type. 1 = PROFIdrive, 0 = Analog Drive.
    technologicalObject.Parameters.Find("Actor.Type").Value = 1;

    //Set the encoder mode
    technologicalObject.Parameters.Find("_Sensor[1].Interface.EncoderConnection").Value =
7;

    //Set axis to data block mode
    technologicalObject.Parameters.Find("_Sensor[1].Interface.DataConnection").Value = 1;

    //Set the tag in the data block. For PD_TEL3 and PD_TEL4 "Encoder1" or "Encoder2".
    technologicalObject.Parameters.Find("_Sensor[1].Interface.DataBlock").Value =
>Data_block_1.Member_of_Type_PD_TEL3";
}
```

Parameters for TO_PositioningAxis and TO_CommandTable

You can find a list of all available variables in SIMATIC STEP 7 S7-1200 Motion Control function manual on the internet (<https://support.industry.siemens.com/cs/ww/en/view/109754206>).

Note

In TIA Portal in the Parameter view of the technology object configuration you can find the column "Name in Openness".

7.19.23.15 S7-1500 Motion Control

Creating and finding TO_OutputCam, TO_CamTrack and TO_MeasuringInput

Requirement

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76).
- A project is open.
See Opening a project (Page 106).
- A S7-1500 PLC is determined in the project.
- A technology object of the type TO_PositioningAxis, TO_SynchronousAxis or TO_ExternalEncoder is determined in the project.

Application

The output cam, cam track and measuring input technology objects are associated with positioning axis, synchronous axis or external encoder technology objects. In order to access an output cam, cam track or measuring input technology object you use the service OutputCamMeasuringInputContainer.

Program code: Creating and finding output cam, cam track and measuring input technology objects

Modify the following program code to create or find an output cam, cam track or measuring input technology object:

```

/*An instance of the technology object under which the TO_OutputCam, TO_CamTrack or
TO_MeasuringInput should be created is already available in the program before*/
private static void CreateFind_OutputcamCamtrackMeasuringinput(TechnologicalInstanceDB
technologyObject)
{
    //Retrieve service OutputCamMeasuringInputContainer
    OutputCamMeasuringInputContainer container =
    technologyObject.GetService<OutputCamMeasuringInputContainer>();
    //Get access to TO_OutputCam / TO_CamTrack container
    TechnologicalInstanceDBComposition outputcamCamtrackContainer = container.OutputCams;

    //Find technology object TO_OutputCam or TO_CamTrack
    TechnologicalInstanceDB outputCam = outputcamCamtrackContainer.Find("OutputCamName");
    TechnologicalInstanceDB camTrack = outputcamCamtrackContainer.Find("CamTrackName");

    //Create new technology object TO_OutputCam or TO_CamTrack
    TechnologicalInstanceDB newOutputCam =
    outputcamCamtrackContainer.Create("NewOutputCamName", "TO_OutputCam",
    new Version(3, 0));
    TechnologicalInstanceDB newCamTrack =
    outputcamCamtrackContainer.Create("NewCamTrackName", "TO_CamTrack", new Version(3, 0));

    //Get access to TO_MeasuringInput container
    TechnologicalInstanceDBComposition measuringInputContainer = container.MeasuringInputs;

    //Find technology object TO_MeasuringInput
    TechnologicalInstanceDB measuringInput =
    measuringInputContainer.Find("MeasuringInputName");

    //Create new technology object TO_MeasuringInput
    TechnologicalInstanceDB newMeasuringInput =
    measuringInputContainer.Create("NewMeasuringInput", "TO_MeasuringInput",
    new Version(3, 0));
}

```

Parameters of S7-1500 Motion Control

Most parameters of S7-1500 Motion Control technology objects are directly mapped to data block tags, but there are also some additional parameters that do not map directly to data blocks. In Openness the directly mapped parameters have the same order as in the "data navigation" in the parameter view of the technology object. After the directly mapped parameters the additional parameters follow in order of the table.

Parameters mapped directly to technology object data block tags

You have access to all technology object data block tags as described in general except of:

- Read-only tags
- Tags of data type VREF
- Tags of "InternalToTrace" structure
- Tags of "ControlPanel" structure

You can find additional information about the directly mapped parameters in the appendix of:

- SIMATIC S7-1500 Motion Control function manual:
<https://support.industry.siemens.com/cs/ww/en/view/109749262> (<https://support.industry.siemens.com/cs/ww/en/view/109749262>)
- SIMATIC S7-1500T Motion Control function manual:
<https://support.industry.siemens.com/cs/ww/en/view/109749263> (<https://support.industry.siemens.com/cs/ww/en/view/109749263>)
- SIMATIC S7-1500T Kinematics Functions function manual:
<https://support.industry.siemens.com/cs/ww/en/view/109749264> (<https://support.industry.siemens.com/cs/ww/en/view/109749264>)

Some technology parameters that map to read-only data block tags need to be made writeable in the PublicAPI. The allowed values are the same ones as for the underlying data block tags. The affected parameters are listed in the following tables:

Name in Openness	Data type	TO_SpeedAxis	TO_PositioningAxis	TO_SynchronousAxis	TO_ExternalEncoder
Actor.Type	int	X	X	X	-
Actor.Interface.EnableDriveOutput	bool	X	X	X	-
Actor.Interface.DriveReadyInput	bool	X	X	X	-
Actor.DataAdaptionOffline	bool	X	X	X	-
VirtualAxis.Mode	uint	X	X	X	-
Sensor[n].DataAdaptionOffline ¹⁾	bool	-	X	X	-
Sensor[n].Existence ¹⁾	bool	-	X	X	-
Sensor[n].Interface.Number ¹⁾	uint	-	X	X	-
Sensor[n].Type ¹⁾	int	-	X	X	-
Sensor.DataAdaptionOffline	bool	-	-	-	X
Sensor.Interface.Number	uint	-	-	-	X
Sensor.Type	int	-	-	-	X

Name in Openness	Data type	TO_OutputCam	TO_MeasuringInput	TO_Kinematics ²⁾
Interface.LogicOperation	int	X	-	-
Parameter.MeasuringInput-Type	int	-	X	-

7.19 Functions for accessing the data of a PLC device

Name in Openness	Data type	TO_OutputCam	TO_MeasuringInput	TO_Kinematics ²⁾
Kinematics.TypeOfKinematics	int	-	-	X
MotionQueue.MaxNumberOfCommands	int	-	-	X

1) S7-1500 PLC: n=1; S7-1500T PLC: 1≤n≤4

2) S7-1500T PLC

Parameters not mapped directly to technology object data block tags

For S7-1500 Motion Control technology objects the following additional parameters which do not directly map to data block tags are available:

Name in Openness	Name in function view	Possible value	Data type in Openness	TO_SpeedAxis	TO_PositioningAxis TO_SynchronousAxis	TO_External-Encoder
_Properties.Motion-Type	Axis type respectively "Technological unit of the position"	0: Linear 1: Rotary	int	-	X	X
_Units.LengthUnit	Position units	See tag Units.Length-Unit ³⁾	uint	-	X	X
_Units.VelocityUnit	Velocity units	See tag Units.VelocityUnit ³⁾	uint	X	X	X
_Units.TorqueUnit	Torque units	See tag Units.TorqueUnit ³⁾	uint	X	X	-
_Units.ForceUnit	Force units	See tag Units.ForceUnit ³⁾	uint	-	X	-
_Actor.Interface.Telegram	Drive telegram	Telegram number ⁴⁾	uint	X	X	-
_Actor.Interface.EnableDriveOutputAddress	Drive output address	PublicAPI-object	SW.Tags.PlcTag	X	X	-
_Actor.Interface.DriveReadyInputAddress	Drive ready input address	PublicAPI-object	SW.Tags.PlcTag	X	X	-
_Sensor[n].Interface.Telegram ⁵⁾	Encoder telegram	Telegram number ⁴⁾	uint	-	X	-
_Sensor[n].ActiveHoming.DigitalInputAddress ⁵⁾	Digital input	PublicAPI-object	SW.Tags.PlcTag	-	X	-
_Sensor[n].PassiveHoming.DigitalInputAddress ⁵⁾	Digital input	PublicAPI-object	SW.Tags.PlcTag	-	X	-

Name in Openness	Name in function view	Possible value	Data type in Openness	TO_SpeedAxis	TO_PositioningAxis TO_SynchronousAxis	TO_ExternalEncoder
_PositionLimits_HW.MinSwitchAddress	Hardware low limit switch input	PublicAPI-object	SW.Tags.PlcTag	-	X	-
_PositionLimits_HW.MaxSwitchAddress	Hardware high limit switch input	PublicAPI-object	SW.Tags.PlcTag	-	X	-
_Sensor.Interface.Telegram	Encoder telegram	Telegram number ⁴⁾	uint	-	-	X
_Sensor.PassiveHoming.DigitalInputAddress	Digital input	PublicAPI-object	SW.Tags.PlcTag	-	-	X

For output cam, cam track and measuring input technology objects the following additional parameter is available:

Name in Openness	Name in function view	Possible value	Data type
_AssociatedObject	Associated object	PublicAPI-object	SW.TechnologicalObjects.TechnologicalInstanceDB

For kinematics technology object the following additional parameters are available (S7-1500T):

Name in Openness	Name in function view	Possible value	Data type
_KinematicsAxis[1...4]	Axis 1 - 3, Orientation axis	Axis that can be connected to TO_Kinematics objects	SW.TechnologicalObjects.TechnologicalInstanceDB
_Units.LengthUnit	Units of measurement > Position	See tag Units.LengthUnit ³⁾	uint
_Units.LengthVelocityUnit	Units of measurement > Velocity	See tag Units.LengthVelocityUnit ³⁾	uint
_Units.AngleUnit	Units of measurement > Angle	See tag UnitsAngleUnit ³⁾	uint
_Units.AngleVelocityUnit	Units of measurement > Angle velocity	See tag Units.AngleVelocityUnit ³⁾	uint

3) possible values are described in the function manual S7-1500 Motion Control on chapter units tags (TO)

4) possible values are described in the function manual S7-1500 Motion Control on chapter PROFdrive telegrams

5) S7-1500 PLC: n=1; S7-1500T PLC: 1≤n≤4

Program code: Directly mapped data block tags

Modify the following program code to access the directly mapped parameters:

```
//An instance of the technology object is already available in the program before
private static void ReadWriteDataBlockTag(TechnologicalInstanceDB technologyObject)
{
    //Read value from data block tag "ReferenceSpeed"
    double value =
        (double)technologyObject.Parameters.Find("Actor.DriveParameter.ReferenceSpeed").Value;

    //Write data block tag "ReferenceSpeed"
    technologyObject.Parameters.Find("Actor.DriveParameter.ReferenceSpeed").Value = 3000.0;
}
```

Program code: Additional parameters

Modify the following program code to access the additional parameters:

```
//An instance of the technology object is already available in the program before
private static void ReadWriteAdditionalParameter(TechnologicalInstanceDB technologyObject)
{
    //Read additional parameter "_Properties.MotionType"
    uint value = (uint)technologyObject.Parameters.Find("_Properties.MotionType").Value;

    //Write additional parameter "_Properties.MotionType"
    technologyObject.Parameters.Find("_Properties.MotionType").Value = 1;
}
```

Additional information

You can find additional information in:

- SIMATIC S7-1500 Motion Control function manual:
<https://support.industry.siemens.com/cs/ww/en/view/109749262> (<https://support.industry.siemens.com/cs/ww/en/view/109749262>)
- SIMATIC S7-1500T Motion Control function manual:
<https://support.industry.siemens.com/cs/ww/en/view/109749263> (<https://support.industry.siemens.com/cs/ww/en/view/109749263>)
- SIMATIC S7-1500T Kinematics Functions function manual:
<https://support.industry.siemens.com/cs/ww/en/view/109749264> (<https://support.industry.siemens.com/cs/ww/en/view/109749264>)

Connecting drives

Requirement

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76).
- A project is open.
See Opening a project (Page 106).
- A S7-1500 PLC is determined in the project.
- A technology object of the type TO_SpeedAxis, TO_PositioningAxis or TO_SynchronousAxis is determined in the project.
- A drive is determined in the project.

Application

To connect an axis with a drive, it is necessary to specify several values together in a single call. The public API type AxisEncoderHardwareConnectionInterface provides the following methods which can be used to connect and disconnect the actor or sensor interfaces:

Method	Description
void Connect(HW.Deviceltem moduleInOut)	Connects to input and output addresses at one module.
void Connect(HW.Deviceltem moduleIn, HW.Deviceltem moduleOut)	Connects to input and output addresses at separate modules.
void Connect(HW.Deviceltem moduleIn, HW.Deviceltem moduleOut, ConnectOption connectOption)	Connects to input and output addresses at separate modules, specifying an additional ConnectOption
void Connect(HW.Channel channel)	Connects to a channel
void Connect(int addressIn, int addressOut, ConnectOption connectOption)	Connects specifying bit addresses directly
void Connect(string pathToDBMember)	Connects to a data block tag
void Connect(SW.Tags.PlcTag outputTag)	Connects to a PLC tag
void Disconnect()	Disconnects an existing connection

Note

Automatic connections

Note that the same behavior as in the user interface also applies here. Whenever the actor interface is connected via one of the connection methods and the telegram contains a sensor part or telegram 750. These parts are connected automatically.

7.19 Functions for accessing the data of a PLC device

You can use the following read-only attributes to determine how the technology object is connected. The respective connection values are set only if the connection of the specific kind exists.

Attribute	Data type	Description
IsConnected	bool	TRUE: Interface is connected FALSE: Interface is not connected
InputOutputModule	HW.DeviceItem	Connected module that contains input and output addresses
InputModule	HW.DeviceItem	Connected module that contains input addresses The value is also set in case of an existing connection to a module containing input and output addresses.
OutputModule	HW.DeviceItem	Connected module that contains output addresses The value is also set in case of an existing connection to a module containing input and output addresses.
InputAddress	int	Logical input address of connected object; for example, 256.
OutputAddress	int	Logical output address of connected object; for example, 256.
ConnectOption	ConnectOption	Value of the ConnectOption that has been set when the connection was made: <ul style="list-style-type: none">• Default Only modules that are recognized as valid connection partners can be selected.• AllowAllModules Corresponds to selecting "Show all modules" in the user interface.
Channel	HW.Channel	Connected channel
PathToDBMember	string	Connected technology object data block tag
OutputTag	SW.Tags.PlcTag	Connected PLC tag (analog connection)
SensorIndexInActor-Telegram	int	Connected sensor part in actor telegram The attribute is only relevant for sensor interfaces. 0: Encoder is not connected 1: Encoder is connected to first sensor interface in telegram 2: Encoder is connected to second sensor interface in telegram For the actor interface the value is always 0.

Note**Access the sensor interface**

To access the sensor interface you can use SensorInterface[m] with $0 \leq m \leq 3$.

Program Code: void Connect(HW.DeviceItem moduleInOut)

Modify the following program code to connect a mixed module that contains input and output addresses:

```
//An instance of technology object and device item is already available in the program  
before  
private static void UseServiceAxisHardwareConnectionProvider(TechnologicalInstanceDB  
technologyObject, DeviceItem devItem)  
{  
    //Retrieve service AxisHardwareConnectionProvider  
    AxisHardwareConnectionProvider connectionProvider =  
    technologyObject.GetService<AxisHardwareConnectionProvider>();  
  
    //Connect ActorInterface with DeviceItem  
    connectionProvider.ActorInterface.Connect(devItem);  
  
    //Connect first SensorInterface with DeviceItem  
    connectionProvider.SensorInterface[0].Connect(devItem);  
  
    //Check ConnectionState of ActorInterface  
    bool actorInterfaceConnectionState = connectionProvider.ActorInterface.IsConnected;  
  
    //Check ConnectionState of first SensorInterface  
    bool sensorInterfaceConnectionState =  
    connectionProvider.SensorInterface[0].IsConnected;  
}
```

Connecting telegram 750**Requirement**

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- A S7-1500 PLC is determined in the project.
- A technology object of the type TO_SpeedAxis, TO_PositioningAxis or TO_SynchronousAxis V4.0 is determined in the project.
- A drive that supports telegram 750 is determined in the project.

Application

If telegram 750 was added after connecting the drive and the axis, it is necessary to connect telegram 750 separately. EnableTorqueData is set to TRUE automatically. The public API type TorqueHardwareConnectionInterface provides the following methods which can be used to connect and disconnect telegram 750:

Method	Description
void Connect(HW.Deviceltem moduleInOut)	Connects to input and output addresses at one module
void Connect(HW.Deviceltem moduleIn, HW.Deviceltem moduleOut)	Connects to input and output addresses at separate modules
void Connect(HW.Deviceltem moduleIn, HW.Deviceltem moduleOut, ConnectOption connectOption)	Connects to input and output addresses at separate modules, specifying an additional ConnectOption
void Connect(int addressIn, int addressOut, ConnectOption connectOption)	Connects specifying bit addresses directly
void Connect(string pathToDBMember)	Connects to a data block tag
void Disconnect()	Disconnects an existing connection

The TorqueHardwareConnectionInterface can be retrieved via the property TorqueInterface at the type AxisHardwareConnectionProvider. If the connection to telegram 750 is not supported, the property value is “null”.

If the drive is connected by data block tags, you cannot connect telegram 750 by module. You can use the following read-only attributes to determine how the technology object is connected. The respective connection values are set only if the connection of the specific kind exists:

Attribute	Data type	Description
IsConnected	bool	TRUE: Interface is connected FALSE: Interface is not connected
InputOutput-Module	HW.Deviceltem	Connected module that contains input and output addresses
InputModule	HW.Deviceltem	Connected module that contains input addresses The value is also set in case of an existing connection to a module containing input and output addresses.
OutputMod-ule	HW.Deviceltem	Connected module that contains output addresses The value is also set in case of an existing connection to a module containing input and output addresses.
InputAddress	int	Logical input address of connected object; for example 256
OutputAd-dress	int	Logical output address of connected object; for example 256

Attribute	Data type	Description
ConnectOption	ConnectOption	Value of the ConnectOption that has been set when the connection was made: <ul style="list-style-type: none">• Default Only modules that are recognized as valid connection partners can be selected.• AllowAllModules Corresponds to selecting "Show all modules" in the user interface.
PathToDB-Member	string	Connected technology object data block tag

Program Code: Connect telegramm 750

Modify the following program code to connect a mixed module that contains input and output addresses:

```
//An instance of technology object and device item is already available in the program
before
private static void ConnectTorqueInterface(TechnologicalInstanceDB technologyObject,
DeviceItem devItem)
{
    //Retrieve service AxisHardwareConnectionProvider
    AxisHardwareConnectionProvider connectionProvider =
    technologyObject.GetService<AxisHardwareConnectionProvider>();
    //Connect TorqueInterface with DeviceItem
    connectionProvider.TorqueInterface.Connect(devItem);
}
```

Connecting encoders

Requirement

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76).
- A project is open.
See Opening a project (Page 106).
- A S7-1500 PLC is determined in the project.
- A technology object of the type TO_ExternalEncoder is determined in the project.
- An object is determined in the project that provides PROFIdrive telegram 81 or 83.

Application

To connect an external encoder technology object with the encoder hardware, it is necessary to specify several values together in a single call. The public API type AxisEncoderHardwareConnectionInterface provides the following methods which can be used to connect and disconnect the sensor interface:

Method	Description
void Connect(HW.Deviceltem moduleInOut)	Connects to input and output addresses at one module.
void Connect(HW.Deviceltem moduleIn, HW.Deviceltem moduleOut)	Connects to input and output addresses at separate modules.
void Connect(HW.Deviceltem moduleIn, HW.Deviceltem moduleOut, ConnectOption connectOption)	Connects to input and output addresses at separate modules, specifying an additional ConnectOption
void Connect(HW.Channel channel)	Connects to a channel
void Connect(int addressIn, int addressOut, ConnectOption connectOption)	Connects specifying bit addresses directly
void Connect(string pathToDBMember)	Connects to a data block tag
void Connect(SW.Tags.PlcTag outputTag)	Not relevant for connecting encoders
void Disconnect()	Disconnects an existing connection

You can use the following read-only attributes to determine how the technology object is connected. The respective connection values are set only if the connection of the specific kind exists.

Attribute	Data type	Description
IsConnected	bool	TRUE: Interface is connected FALSE: Interface is not connected
InputOutputModule	HW.Deviceltem	Connected module that contains input and output addresses
InputModule	HW.Deviceltem	Connected module that contains input addresses The value is also set in case of an existing connection to a module containing input and output addresses.
OutputModule	HW.Deviceltem	Connected module that contains output addresses The value is also set in case of an existing connection to a module containing input and output addresses.
InputAddress	int	Logical input address of connected object, for example 256.
OutputAddress	int	Logical output address of connected object, for example 256.
ConnectOption	ConnectOption	Value of the ConnectOption that has been set when the connection was made: <ul style="list-style-type: none">• Default Only modules that are recognized as valid connection partners can be selected.• AllowAllModules Corresponds to selecting "Show all modules" in the user interface.
Channel	HW.Channel	Connected channel
PathToDBMember	string	Connected data block tag

Attribute	Data type	Description
OutputTag	SW.Tags.PlcTag	Not relevant for connecting encoders
SensorIndexInActor-Telegram	int	<p>Connected sensor telegram</p> <p>The attribute is only relevant for sensor interfaces.</p> <p>0: Encoder is not connected</p> <p>1: Encoder is connected to first sensor interface in telegram</p> <p>2: Encoder is connected to second sensor interface in telegram</p> <p>For the actor interface the value is always 0.</p>

Program code: Connect an encoder

Modify the following program code to connect an external encoder technology object:

```
//An instance of technology object and device item is already available in the program before
private static void UseServiceEncoderHardwareConnectionProvider(TechnologicalInstanceDB
technologyObject, DeviceItem devItem)
{
    //Retrieve service EncoderHardwareConnectionProvider
    EncoderHardwareConnectionProvider connectionProvider =
    technologyObject.GetService<EncoderHardwareConnectionProvider>();

    //Connect SensorInterface with DeviceItem
    connectionProvider.SensorInterface.Connect(devItem);

    //Check ConnectionState of SensorInterface
    bool sensorInterfaceConnectionState = connectionProvider.SensorInterface.IsConnected;
}
```

Connecting output cams and cam tracks to hardware

Requirement

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76).
- A project is open.
See Opening a project (Page 106).
- A S7-1500 PLC is determined in the project.
- A technology object of the type TO_OutputCam or TO_CamTrack is determined in the project.
- A digital output module is determined in the project, for example TM Timer DIDQ.

Application

To connect an output cam or cam track technology object with a digital output, it is necessary to specify several values together in a single call. The public API type OutputCamHardwareConnectionProvider provides the following methods which can be used to connect and disconnect the actor or sensor interfaces:

Method	Description
void Connect(HW.Channel channel)	Connects to a channel
void Connect(SW.Tags.PlcTag outputTag)	Connects to a PLC tag
void Connect(int address)	Connects specifying bit addresses directly
void Disconnect()	Disconnects an existing connection

You can use the following read-only attributes to determine how the technology object is connected:

Attribute	Data type	Description
IsConnected	bool	TRUE: Technology object is connected FALSE: Technology object is not connected
Channel	HW.Channel	Connected channel
OutputTag	SW.Tags.PlcTag	Connected PLC tag
OutputAddress	int	Logical output address of connected object, for example 256.

Program code: Connect output cam or cam track technology object

Modify the following program code to connect an output cam or cam track technology object:

```
//An instance of technology object and channel item is already available in the program
before
private static void UseServiceOutputCamHardwareConnectionProvider(TechnologicalInstanceDB
technologyObject, Channel channel)

{
    //Retrieve service OutputCamHardwareConnectionProvider
    OutputCamHardwareConnectionProvider connectionProvider =
    technologyObject.GetService<OutputCamHardwareConnectionProvider>();

    //Connect technology object with Channel
    connectionProvider.Connect(channel);

    //Check ConnectionState of technology object
    bool connectionState = connectionProvider.IsConnected;
}
```

Connecting measuring inputs to hardware

Requirement

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76).
- A project is open.
See Opening a project (Page 106).
- A S7-1500 PLC is determined in the project.
- A technology object of the type TO_MeasuringInput is determined in the project.
- A digital input module is determined at drive or in the project, for example TM Timer DIDQ.

Application

To connect a measuring input technology object with a digital input, it is necessary to specify several values together in a single call. The public API type MeasuringInputHardwareConnectionProvider provides the following methods which can be used to connect and disconnect the actor or sensor interface:

Method	Description
void Connect(HW.Channel channel)	Connects to a channel
void Connect(HW.Deviceltem moduleIn, int channelIndex)	Connects to a module, specifying an additional channel index
void Connect(int address)	Connects specifying bit addresses directly
void Disconnect()	Disconnects an existing connection

You can use the following read-only attributes to determine how the technology object is connected:

Attribute	Data type	Description
IsConnected	bool	TRUE: Technology object is connected FALSE: Technology object is not connected
InputModule	HW.Deviceltem	Connected module that contains input addresses
ChannelIndex	int	Index of connected channel with respect to InputModule
Channel	HW.Channel	Connected channel
InputAddress	int	Logical input address of connected object, for example 256.

Program code: Connect a measuring input technology object

Modify the following program code to connect a measuring input technology object:

```
//An instance of technology object and channel item is already available in the program
before
private static void
UseServiceMeasuringInputHardwareConnectionProvider(TechnologicalInstanceDB
technologyObject, Channel channel)
{
    //Retrieve service MeasuringInputHardwareConnectionProvider
    MeasuringInputHardwareConnectionProvider connectionProvider =
    technologyObject.GetService<MeasuringInputHardwareConnectionProvider>();

    //Connect technology object with Channel
    connectionProvider.Connect(channel);

    //Check ConnectionState of technology object
    bool connectionState = connectionProvider.IsConnected;
}
```

Connecting synchronous axis with leading values

Requirement

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76).
- A project is open.
See Opening a project (Page 106).
- A S7-1500 PLC is determined in the project.
- A technology object of the type TO_PositioningAxis, TO_SynchronousAxis or TO_ExternalEncoder as leading axis is determined in the project.
- A technology object of the type TO_SynchronousAxis as following axis is determined in the project.

Application

To connect a synchronous axis technology object with leading values, it is necessary to specify several values together in a single call. The public API type SynchronousAxisMasterValues provides the following methods which can be used to connect and disconnect leading values. Leading values can be connected as setpoint coupling (S7-1500 PLC, S7-1500T PLC) or actual value coupling (S7-1500T PLC). All methods and attributes are relevant for both types of coupling.

Method	Description
int IndexOf (TechnologicalInstanceDB element)	Returns the corresponding index of a leading value
bool Contains (TechnologicalInstanceDB element)	TRUE: The container contains the leading value FALSE: The container does not contain the leading value

Method	Description
IEnumerator <TechnologicalInstanceDB>()	Used to support each iteration
void Add (TechnologicalInstanceDB element)	Connects following axis to leading value
bool Remove (TechnologicalInstanceDB element)	Disconnects following axis from leading value TRUE: Disconnection was successful FALSE: Disconnection was not successful

You can use the following read-only attributes:

Attribute	Data type	Description
Count	int	Count of leading values
IsReadonly	bool	TRUE: The container is read-only FALSE: The container is not read-only
Parent	IEngineeringObject	Returns the parent of the container. In this case parent means the service SynchronousAxisMasterValues.
this [id] { get; }	TechnologicalInstanceDB	Index-based access to leading values

Program code: Connect a synchronous axis with a leading value

Modify the following program code to connect a synchronous axis with a leading value:

```
//An instance of leading axis and following axis is already available in the program before
private static void UseServiceSynchronousAxisMasterValues(TechnologicalInstanceDB
masterTechnologyObject, TechnologicalInstanceDB synchronousTechnologyObject)
{
    //Retrieve service SynchronousAxisMasterValues
    SynchronousAxisMasterValues masterValues =
        synchronousTechnologyObject.GetService<SynchronousAxisMasterValues>();

    //Connect following axis and leading axis with setpoint coupling
    masterValues.SetPointCoupling.Add(masterTechnologyObject);

    //Get container of connected leading axis with setpoint coupling
    TechnologicalInstanceDBAssociation setPointMasterValues =
        masterValues.SetPointCoupling;

    //Remove connected leading axis with setpoint coupling
    masterValues.SetPointCoupling.Remove(masterTechnologyObject);

    //Connect following axis and leading axis with actual value coupling
    masterValues.ActualValueCoupling.Add(masterTechnologyObject);

    //Get container of connected leading axis with actual value coupling
    TechnologicalInstanceDBAssociation actualValueMasterValues =
        masterValues.ActualValueCoupling;

    //Remove connected leading axis with actual value coupling
    masterValues.ActualValueCoupling.Remove(masterTechnologyObject);
}
```

Exporting and importing technology object cam (S7-1500T)

Requirement

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76).
- A project is open.
See Opening a project (Page 106).
- A S7-1500 PLC is determined in the project.
See Querying PLC and HMI targets (Page 178)
- The technology object exists.

Application

To export or import the data of a technology object cam you have to specify the format and which separator should be used. The public API type CamDataSupport provides the following methods which can be used to export the data of technology object cam.

Method	Description
void SaveCamDataBinary(System.IO.FileInfo destinationFile)	Exports the data in binary format in the destination file.
void SaveCamDataPointList(System.IO.FileInfo destinationFile, CamDataFormatSeparator separator, int samplePoints)	Exports the data in format "PointList" in the destination file.
void SaveCamData(System.IO.FileInfo destinationFile, CamDataFormat format, CamDataFormatSeparator separator)	Exports the data in the destination file. You can specify data format as "MCD", "SCOUT" or "Pointlist" and separator as "tab" or "comma". If you choose "PointList" 360 interpolation points will be exported.
void LoadCamData(System.IO.FileInfo sourceFile, CamDataFormatSeparator separator)	Imports the cam data in the format "MCD", "SCOUT" or "Pointlist" to the project.
void LoadCamDataBinary(System.IO.FileInfo sourceFile)	Imports the cam data from a binary file to the project.

You can use the following attributes:

Attribute	Data type	Description
separator	CamDataFormatSeparator	Allowed values <ul style="list-style-type: none"> • tab • comma
samplePoints	int	Number of interpolation points that should be exported.
format	CamDataFormat	Allowed values <ul style="list-style-type: none"> • MCD • SCOUT • Pointlist

Attribute	Data type	Description
destinationFile	System.IO.FileInfo	Name of destination file. Must not be null. Access rights and enough space on storage medium must be given. An existing file will be overwritten.
sourceFile	System.IO.FileInfo	Name of source file. Must not be null. Access rights must be given. Content must be in specified format.

Program code: Export cam data

Modify the following program code to export cam data:

```
//An instance of technology object is already available in the program before
private static void ExportCamData(TechnologicalInstanceDB technologyObject,
System.IO.FileInfo destinationFile)
{
    //Retrieve service CamDataSupport
    CamDataSupport camData = technologyObject.GetService<CamDataSupport>();

    //Save cam data in MCD format, using the separator Tab
    camData.SaveCamData(destinationFile, CamDataFormat.MCD, CamDataFormatSeparator.Tab);
}
```

Program code: Import cam data

Modify the following program code to import cam data:

```
//An instance of technology object is already available in the program before
private static void ImportCamData(TechnologicalInstanceDB technologyObject,
System.IO.FileInfo sourceFile)
{
    //Retrieve service CamDataSupport
    CamDataSupport camData = technologyObject.GetService<CamDataSupport>();

    //Load cam data from source file, using the separator Tab
    camData.LoadCamData(sourceFile, CamDataFormatSeparator.Tab);
}
```

7.19.23.16 PID control

Parameters for PID_Compact, PID_3Step, PID_Temp, CONT_C, CONT_S, TCONT_CP and TCONT_S

You can find a list of all available parameters in the product information “Parameters of technology objects in TIA Portal Openness“ on the internet (<https://support.industry.siemens.com/cs/ww/en/view/109744932>).

For each parameter the following properties are provided:

- Name in configuration (TIA Portal)
- Name in Openness

- Data type in Openness
- Default access
- Range of values

Note

In TIA Portal in the Parameter view of the technology object configuration you can find the column "Name in Openness".

Additional information

You can find additional information in SIMATIC S7-1200/S7-1500 PID control function manual on the internet (<https://support.industry.siemens.com/cs/ww/en/view/108210036>).

7.19.23.17 Counting

Parameters for High_Speed_Counter and SSI_Absolute_Encoder

You can find a list of all available parameters in the product information "Parameters of technology objects in TIA Portal Openness" on the internet (<https://support.industry.siemens.com/cs/ww/en/view/109744932>).

For each parameter the following properties are provided:

- Name in configuration (TIA Portal)
- Name in Openness
- Data type in Openness
- Default access
- Range of values

Additional information

You can find additional information in SIMATIC S7-1500, ET 200MP, ET 200SP Counting, measurement and position input function manual on the internet (<http://support.automation.siemens.com/WW/view/en/59709820>).

7.19.23.18 Easy Motion Control

Parameters for AXIS_REF

You can find a list of all available parameters in the product information "Parameters of technology objects in TIA Portal Openness" on the internet (<https://support.industry.siemens.com/cs/ww/en/view/109744932>).

For each parameter the following properties are provided:

- Name in configuration (TIA Portal)
- Name in Openness
- Data type in Openness
- Default access
- Range of values

Note

In TIA Portal in the Parameter view of the technology object configuration you can find the column "Name in Openness".

Additional information

You can find additional information for Easy Motion Control in the Information system of STEP 7 (TIA Portal).

7.19.24 Tags and Tag tables

7.19.24.1 Starting the "PLC Tags" editor

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- Instance of the TIA Portal is opened with user interface.

Program code

Modify the following program code to start the corresponding editor for an object reference of the type `PlcTagTable` in the TIA Portal instance:

```
//Opens tagtable in editor "Tags"
private static void OpenTagtableInEditor(PlcSoftware plcSoftware)
{
    PlcTagTable plcTagTable = plcSoftware.TagTableGroup.TagTables.Find("MyTagTable");
    plcTagTable.ShowInEditor();
}
```

See also

[Importing configuration data \(Page 753\)](#)

7.19.24.2 Querying system groups for PLC tags

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)
- A PlcSoftware instance was retrieved from a PLC device item.
See [Querying PLC and HMI targets \(Page 178\)](#)

Program code

Modify the following program code to query the system group for PLC tags:

```
//Retrieves the plc tag table group from a plc
private PlcTagTableSystemGroup GetControllerTagfolder(PlcSoftware plcSoftware)
{
    PlcTagTableSystemGroup plcTagTableSystemGroup = plcSoftware.TagTableGroup;
    return plcTagTableSystemGroup;
}
```

7.19.24.3 Creating PLC tag table

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)
- A PlcSoftware instance was retrieved from a PLC device item.
See [Querying PLC and HMI targets \(Page 178\)](#)

Program code

Modify the following program code to create the PLC tag table. It creates a new tag table with the given name in the composition.

```
PlcTagTable myTable = plc.TagTableGroup.TagTables.Create("myTable");
```

See also

[Querying PLC and HMI targets \(Page 178\)](#)

7.19.24.4 Enumerating user-defined groups for PLC tags

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)
- A PlcSoftware instance was retrieved from a PLC device item.
See [Querying PLC and HMI targets \(Page 178\)](#)

Application

Subfolders are taken into account recursively for enumeration.

Program code: Enumerating user-defined groups for PLC tags

Modify the following program code to enumerate user-defined groups for PLC tags:

```
//Enumerates all plc tag table user groups including subgroups
private static void EnumeratePlcTagTableUserGroups(PlcSoftware plcSoftware)
{
    foreach (PlcTagTableUserGroup plcTagTableUsergroup in plcSoftware.TagTableGroup.Groups)
    {
        EnumerateTagTableUserGroups(plcTagTableUsergroup);
    }
}
private static void EnumerateTagTableUserGroups(PlcTagTableUserGroup tagTableUsergroup)
{
    foreach (PlcTagTableUserGroup plcTagTableUsergroup in tagTableUsergroup.Groups)
    {
        EnumerateTagTableUserGroups(plcTagTableUsergroup);
        // recursion
    }
}
```

Program code: Accessing a user-defined group

Modify the following program code to access a user-defined group for PLC tags:

```
//Gives individual access to a specific plc tag table user folder
private static void AccessPlcTagTableUserGroupWithFind(PlcSoftware plcSoftware, string
folderToFind)
{
    PlcTagTableUserGroupComposition plcTagTableUserGroupComposition =
plcSoftware.TagTableGroup.Groups;
    PlcTagTableUserGroup controllerTagUserFolder =
plcTagTableUserGroupComposition.Find(folderToFind);
    // The parameter specifies the name of the user folder
}
```

7.19.24.5 Creating user-defined groups for PLC tags

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The TIA Portal Openness API interface supports the creation of a user-defined group for PLC tags.

Program code

Modify the following program code to create a user-defined group for PLC tags:

```
//Creates a plc tag table user group
private static void CreatePlcTagTableUserGroup(PlcSoftware plcSoftware)
{
    PlcTagTableSystemGroup systemGroup = plcSoftware.TagTableGroup;
    PlcTagTableUserGroupComposition groupComposition = systemGroup.Groups;
    PlcTagTableUserGroup myCreatedGroup = groupComposition.Create("MySubGroupName");
    // Optional;
    // create a subgroup
    PlcTagTableUserGroup mySubCreatedGroup =
myCreatedGroup.Groups.Create("MySubSubGroupName");
}
```

7.19.24.6 Deleting user-defined groups for PLC tags

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The TIA Portal Openness API interface supports the deletion of a specific user-defined group for PLC tag tables.

Program code

Modify the following program code to delete a specific user-defined group for PLC tag tables:

```
private static void DeletePlcTagTableUserGroup(PlcSoftware plcSoftware)
{
    PlcTagTableUserGroup group = plcSoftware.TagTableGroup.Groups.Find("MySubGroupName");
    if (group != null)
    {
        group.Delete();
    }
}
```

7.19.24.7 Enumerating PLC tag tables in a folder

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Program code: Enumerating PLC tag tables

Modify the following program code to enumerate all PLC tag tables in system groups or in user-defined groups:

```
//Enumerates all plc tag tables in a specific system group or and user group
private static void EnumerateAllPlcTagTablesInFolder(PlcSoftware plcSoftware)
{
    PlcTagTableComposition tagTables = plcSoftware.TagTableGroup.TagTables;
    // alternatively, PlcTagTableComposition tagTables =
plcSoftware.TagTableGroup.Groups.Find("UserGroup XYZ").TagTables;
    foreach (PlcTagTable tagTable in tagTables)
    {
        // add code here
    }
}
```

Program code: Accessing PLC tag table

Modify the following program code to access the PLC tag table:

```
//Gives individual access to a specific Plc tag table
private static void AccessToPlcTagTableWithFind(PlcSoftware plcSoftware)
{
    PlcTagTableComposition tagTables = plcSoftware.TagTableGroup.TagTables;
    // alternatively, PlcTagTableComposition tagTables =
plcSoftware.TagTableGroup.Groups.Find("UserGroup XYZ").TagTables;
    PlcTagTable controllerTagTable = tagTables.Find("Tag table XYZ");
    // The parameter specifies the name of the tag table
}
```

7.19.24.8 Querying information from a PLC tag table

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

Via PLC tag tables you can access user constants, system constants and tags. The count of the tag composition of a tag table is equal to the number of tags in that tag table. The PLCTagTable contains the following navigators, attributes, and actions.

The following attributes are accessed in PLC tag table.

Name	Type	Type
IsDefault	bool	Read-only
ModifiedTimeStamp	DateTime	Read-only
Name	string	Read-only

The PLCTag table contains the following actions as given below.

Name	Return type	Description
Delete	void	Deletes the instance. Throws an exception if IsDefault is true.
Export	void	Exports the Simatic ML of a Plc tag table.
ShowInEditor	void	Shows the tag table in the Plc tag table editor.

Program code

Modify the following program code to query the information for a PLC tag table:

```

private static void AccessPlcConstantsUsingFind(PlcTagTable tagTable)
{
    PlcUserConstantComposition plcUserConstants = tagTable.UserConstants;
    PlcUserConstant plcUserConstant = plcUserConstants.Find("Constant XYZ");
    //PlcSystemConstantComposition plcSystemConstants = tagTable.SystemConstants;
    //PlcSystemConstant plcSystemConstant = plcSystemConstants.Find("Constant XYZ");
}
private static void EnumeratePlcTags(PlcTagTable tagTable)
{
    PlcTagComposition plcTags = tagTable.Tags;
    foreach (PlcTag plcTag in plcTags)
    {
        string name = plcTag.Name; string typeName = plcTag.DataTypeName;
        string logicalAddress = plcTag.LogicalAddress;
    }
}
private static void EnumeratePlcTagsUsingFind(PlcTagTable tagTable)
{
    PlcTagComposition plcTags = tagTable.Tags;
    PlcTag plcTag = plcTags.Find("Constant XYZ");
}

```

7.19.24.9 Reading the time of the last changes of a PLC tag table

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The format of the time stamp is UTC.

Program code

Modify the following program code to read the time stamp of a specific PLC tag table:

```
//Reads Time-Stamp of a plc Tag Table
private static void GetLastModificationDateOfTagtable(PlcSoftware plcSoftware)
{
    PlcTagTable plcTagTable = plcSoftware.TagTableGroup.TagTables.Find("MyTagTable");
    DateTime modifiedTagTableTimeStamp = plcTagTable.ModifiedTimeStamp;
}
```

7.19.24.10 Deleting a PLC tag table from a group

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Program code

Modify the following program code to delete a specific tag table from a group:

```
//Deletes a PlcTagTable of a group
private static void DeletePlcTagTableInAGroup(PlcSoftware plcSoftware)
{
    PlcTagTableSystemGroup group = plcSoftware.TagTableGroup;
    PlcTagTable tagtable = group.TagTables.Find("MyTagTable");
    if (tagtable!= null)
    {
        tagtable.Delete();
    }
}
```

7.19.24.11 Enumerating PLC tags

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Program code: Enumerating PLC tags in tag tables

Modify the following program code to enumerate all PLC tags in a tag table:

```
//Enumerates all plc tags in a specific tag table
private static void EnumerateAllPlcTagsInTagTable(PlcSoftware plcSoftware)
{
    PlcTagTable tagTable = plcSoftware.TagTableGroup.TagTables.Find("Tagtable XYZ");
    foreach (PlcTag tag in tagTable.Tags)
    {
        // add code here
    }
}
```

7.19.24.12 Accessing PLC tags

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The type `PlcTagComposition` represents a collection of plc tags.

Program code: Accessing a specific PLC tag

Modify the following program code to access the required PLC tag. You have access to the following attributes:

- Name (read only)
- Data type name
- Logical address
- Comment
- ExternalAccessible
- ExternalVisible
- ExternalWritable

```
//Gives individual access to a specific plc tag
private static void AccessPlcTag(PlcTagTable tagTable)
{
    PlcTag tag = tagTable.Tags.Find("Tag XYZ");
    // The parameter specifies the name of the tag
}
```

Program code: Creating tags

Modify the following program code:

```
private static void CreateTagInPLCTagtable(PlcSoftware plcsoftware)
// Create a tag in a tag table with default attributes
{
    string tagName = "MyTag";
    PlcTagTable table = plcsoftware.TagTableGroup.TagTables.Find("myTagTable");
    PlcTagComposition tagComposition = table.Tags;
    PlcTag tag = tagComposition.Create(tagName);
}
```

Modify the following program code:

```
private static void CreateTagInPLCTagtable(PlcSoftware plcsoftware)
// Create a tag of data type bool and logical address not set
{
    string tagName = "MyTag";
    string dataType = "Bool";
    string logicalAddress = "";
    PlcTagTable table = plcsoftware.TagTableGroup.TagTables.Find("myTagTable");
    PlcTagComposition tagComposition = table.Tags;
    PlcTag tag = tagComposition.Create(tagName, dataType, logicalAddress);
}
```

Program code: Deleting tags

Modify the following program code:

```
private static void DeleteTagFromPLCTagtable(PlcSoftware plcsoftware)
// Deletes a single tag of a tag table
{
    string tagName = "MyTag";
    PlcTagTable table = plcsoftware.TagTableGroup.TagTables.Find("myTagTable");
    PlcTagComposition tagComposition = table.Tags;
    PlcTag tag = tagComposition.Find(tagName);
    if (tag != null)
    {
        tag.Delete();
    }
}
```

7.19.24.13 Accessing PLC constants

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The type `PlcUserConstantComposition` represents a collection of plc user constants. You have access to the following attributes:

- Name (read only)
- Data type name
- Value

The type `PlcSystemConstantComposition` represents a collection of plc system constants. You have access to the following attributes:

- Name (read only)
- Data type name (read only)
- Value (read only)

Program code: Creating user constants

Modify the following program code:

```
private static void CreateUserConstantInPLCTagtable(PlcSoftware plcsoftware)
// Create a user constant in a tag table
{
    string constantName = "MyConstant";
    PlcTagTable table = plcsoftware.TagTableGroup.TagTables.Find("myTagTable");
    PlcUserConstantComposition userConstantComposition = table.UserConstants;
    PlcUserConstant userConstant = userConstantComposition.Create(constantName);
}
```

Program code: Deleting user constants

Modify the following program code:

```
private static void DeleteUserConstantFromPLCTagtable(PlcSoftware plcsoftware)
// Deletes a single user constant of a tag table
{
    PlcTagTable table = plcsoftware.TagTableGroup.TagTables.Find("myTagTable");
    PlcUserConstantComposition userConstantComposition = table.UserConstants;
    PlcUserConstant userConstant = userConstantComposition.Find("MyConstant");
    if (userConstant != null)
    {
        userConstant.Delete();
    }
}
```

Program code: Accessing system constants

Modify the following program code:

```
//Gives individual access to a specific system constant
private static void AccessSystemConstant(PlcTagTable tagTable)
{
    PlcTag systemConstant = tagTable.SystemConstants.Find("Constant XYZ");
    // The parameter specifies the name of the tag
}
```

See also

- Creating user-defined groups for PLC tags (Page 568)
- Deleting user-defined groups for PLC tags (Page 569)
- Deleting a PLC tag table from a group (Page 572)
- Accessing PLC tags (Page 573)
- Starting the "PLC Tags" editor (Page 565)
- Reading the time of the last changes of a PLC tag table (Page 572)

7.19.25 Functions for software units

7.19.25.1 Working with software unit

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The units are important parts of the PLC's Programming and can be found on the PlcSoftware.

The unit provider is retrieved from PlcSoftware via GetService and the PlcUnitComposition can be retrieved from it.

You can perform the following elementary operations for units while using TIA Portal Openness:

- Creating units
- Deleting units
- Renaming units

Program code: Creating units

Modify the following program code to create units.

```
PlcUnitProvider provider = m_Target.GetService<PlcUnitProvider>();  
PlcUnitComposition unitComposition = provider.UnitGroup.Units;  
//Creating Unit  
PlcUnit unit1 = unitComposition.Create("Unit1");
```

Error will occur if you provide the wrong name, for example starting with whitespace or containing invalid character or name is too long. A Recoverable Exception will be thrown with

the following error message: "The value of attribute 'Name' contains an invalid character at Position 0."

```
try
{
PlcUnit unit1 = unitComposition.Create("Unit1");
}
catch (EngineeringTargetInvocationException e)
{
Console.WriteLine(e.Message);
}
```

Error will occur if you want to create a unit with which a name already exists. A Recoverable Exception will be thrown with the following message: "Unit with name'Unit1' already exists.

```
PlcUnit unit1 = unitComposition.Create("Unit1");
...
...
try
{
PlcUnit unit2 = unitComposition.Create("Unit1");
}
catch (EngineeringTargetInvocationException e)
{
Console.WriteLine(e.Message);
}
```

Program code: Deleting units

Modify the following program code to delete units.

```
unitProvider.UnitGroup.Units.Find("Unit_2").Delete();
```

You will encounter NullReferenceException in case of the Find() method from above example code does not provide a reference to a Unit.

Program code: Renaming units

You can set the attributes in several ways. It can be adjusted via value assignment or call of SetAttribute as well as SetAttributes. The Name property can be validated by GetAttribute or GetAttributes. To be able to use SetAttribute or GetAttribute the object should be casted to IEngineeringObject.

Modify the following program code to rename units.

```
//Set Value
PlcUnit unit1 = unitComposition.Create("Unit1");
unit1.Name = "Unit1_new";
//Using SetAttributes():
PlcUnit unit1 = unitComposition.Create("Unit1");
var attrList = new List<KeyValuePair<string, object>>()
{
new KeyValuePair<string, object>("Name", "Unit1_new")
};
unit1.SetAttributes(attrList);
//Using SetAttribute()
PlcUnit unit2 = unitComposition.Create("Unit2");
IEngineeringObject unit = (IEngineeringObject)unit2;
unit.SetAttribute("Name", "Unit2_new");
```

Error will occur if you provide wrong name, for example starting with whitespace, or containing invalid character. A Recoverable Exception will be thrown with the error message: "The value of attribute 'Name' contains an invalid character at Position 0."

```
PlcUnit unit1 = unitComposition.Create("Unit_1");
try
{
unit1.Name = "Unit_1";
}
catch (EngineeringTargetInvocationException e)
{
Console.WriteLine(e.Message);
}
```

Error will occur if you want to rename a unit with a name which already exists. A recoverable exception will be thrown with the error message: "The property 'Name' has an invalid value: 'Unit1'. A software unit with the same name already exists."

```
PlcUnit unit1 = unitComposition.Create("Unit1");
PlcUnit unit2 = unitComposition.Create("Unit_2");
try
{
PlcUnit unit2 = unitComposition.Create("Unit1");
}
catch (EngineeringTargetInvocationException e)
{
Console.WriteLine(e.Message);
}
```

See also

[Opening a project \(Page 106\)](#)

7.19.25.2 Accessing software unit

Requirement

- The TIA Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)

Application

You can use the TIA Portal Openness to access the units. The units are important parts of the PLC's programming. Only a selected set of PLCs are supporting units therefore units are not modelled statically in the object model so they aren't browsable directly under the PLC software container.

The PlcUnitProvider makes units accessible and it is available from the PlcSoftware through GetService() – as it shown in the Program code example below:

The PlcUnitProvider allows the access via the UnitGroup navigator to the PlcUnitSystemGroup which contains the PlcUnitComposition with the general composition specific possibilities (iterate, Find, Create, Import etc. for PlcUnits).

The unit's representation (PlcUnit) contains the unit's own specific properties as Openness attributes.

You can access the below attributes and Methods for Openness unit properties:

Attribute Name	Type
Author	String
Name	String

Method Name	Return type
Export	void
Delete	void
GetAttributes	IList<object>
SetAttributes	void

On the PlcUnit, You can iterate through the following objects:

- Comment: MultilingualText
- the BlockGroup (PlcBlockSystemGroup)
 - unit.BlockGroup
- TagTableGroup (PlcTagTableSystemGroup)
 - unit.TagTableGroup
- TypeGroup (PlcTypeSystemGroup)
 - unit.TypeGroup

Program code: Accessing units

```
PlcSoftware plcTarget = GetControllerTargetByPLCName(Session.OpnsProject.Devices, PLCName);
PlcUnitProvider unitProvider = plcTarget.GetService<PlcUnitProvider>();
PlcUnit unit = unitProvider.UnitGroup.Units.Find("Unit_2");
```

See also

- Connecting to the TIA Portal (Page 76)
- Opening a project (Page 106)

7.19.25.3 Accessing software unit underlying objects

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

You can use the TIA Portal Openness to access:

- the Program blocks system group, its contained groups and blocks recursively
- the PLC data types system group, its contained groups and data types recursively
- the PLC tags system group, its contained groups and tag tables recursively

Program code: Accessing program block and block groups

Modify the following program code to access the program block under the current PLC unit by retrieving the program block composition and iterating through its contained block:

```
PlcBlockComposition blockComposition = m_PlcUnit.BlockGroup.Blocks;
foreach (PlcBlock block in blockComposition)
{
...
//usage of block
...
}
```

7.19 Functions for accessing the data of a PLC device

Modify the following program code to access the program block group under the current PLC unit by retrieving the program block group composition, and iterating through its contained groups:

```
PlcBlockUserGroupComposition usergroupComposition = m_PlcUnit.BlockGroup.Groups;
foreach (PlcBlockUserGroup group in usergroupComposition)
{
    PlcBlockComposition blockComposition = group.Blocks;
    foreach (PlcBlock block in blockComposition)
    {
        ...
        //usage of the block
        ...
    }
}
```

Program code: Accessing PLC data types and data type groups

Modify the following program code to access PLC data types under the current PLC unit by retrieving the PLC data type composition and iterating through its contained data types:

```
PlcTypeComposition typeComposition = m_PlcUnit.TypeGroup.Types;
foreach (PlcType type in typeComposition)
{
    ...
    // usage of the type
    ...
}
```

Modify the following program code to access PLC data type group under the current PLC unit by retrieving the PLC data type composition and iterating through its contained groups:

```
PlcTypeUserGroupComposition usergroupComposition = m_PlcUnit.TypeGroup.Groups;
foreach (PlcTypeUserGroup group in usergroupComposition)
{
    PlcTypeCompostion typeComposition = group.Types;
    foreach (PlcType type in typeComposition)
    {
        ...
        // usage of the type
        ...
    }
}
```

Program code: Accessing PLC tag tables and tag table groups

Modify the following program code to access the PLC tag tables under the current PLC unit by retrieving the PLC tag table composition and iterating through its contained tag tables:

```
PlcTagTableComposition tagtableComposition = m_PlatformUnit.TagTableGroup.TagTables;
foreach (PlcTagTable type in tagtableComposition)
{
...
// usage of the tag table
...
}
```

Modify the following program code to access PLC tag table groups under the current PLC unit by retrieving the PLC tag table group composition and iterating through its contained groups:

```
PlcTagTableUserGroupComposition usergroupComposition = m_PlatformUnit.TagTableGroup.Groups;
foreach (PlcTagTableUserGroup group in usergroupComposition)
{
    PlcTagTableComposition tagtableComposition = group.TagTables;
    foreach (PlcTagTable type in tagtableComposition)
    {
...
// usage of the tag table
...
    }
}
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.19.25.4 Accessing to existing relations of a unit

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)

Application

You can use the TIA Portal Openness to access all the existing relations of a unit so that you can read relation properties.

The following attributes are supported by the unit relation in the Openness:

Attribute name	Type	description
RelationType	UnitRelationType	Type of the relation
RelatedObject	String	Contains the name of the accessible element

The following ENUM values are provided for the attribute RelationType:

- Software Unit
- Non-Unit DB
- TO DB

Program code: Accessing relations

You can get relations, its relation type and the name of the related object in several ways.

Modify the following program code for getting the relation composition:

```
PlcSoftware plcTarget = GetControllerTargetByPLCName(Session.OpnsProject.Devices,
PLCName);
PlcUnitProvider provider = plcTarget.GetService<PlcUnitProvider>();
//assuming existing units
m_PlcUnit = provider.UnitGroup.Units[0];
PlcUnitRelationComposition unitRelations = m_PlcUnit.Relations;
```

Modify the following program code for getting relation by indexer:

```
PlcSoftware plcTarget = GetControllerTargetByPLCName(Session.OpnsProject.Devices, PLCName);
PlcUnitProvider provider = plcTarget.GetService<PlcUnitProvider>();
m_PlcUnit = provider.UnitGroup.Units[0]; //assuming existing units
PlcUnitRelation unitRelation = m_PlcUnit.Relations[1];
```

Modify the following program code for getting relations by iterating through them:

```
PlcSoftware plcTarget = GetControllerTargetByPLCName(Session.OpnsProject.Devices, PLCName);
PlcUnitProvider provider = plcTarget.GetService<PlcUnitProvider>();
//assuming existing units
m_PlcUnit = provider.UnitGroup.Units[0];
PlcUnitRelationComposition unitRelations = m_PlcUnit.Relations;
foreach (PlcUnitRelation relation in unitRelations)
{
// using 'relation'
}
```

Modify the following program code for getting the relation type of a unit by the RelationType attribute:

```
PlcSoftware plcTarget =
GetControllerTargetByPLCName(Session.OpnsProject.Devices, PLCName);
PlcUnitProvider provider = plcTarget.GetService<PlcUnitProvider>();
//assuming existing units
m_PlcUnit = provider.UnitGroup.Units[0];
UnitRelationType unitRelationType = m_PlcUnit.Relations[2].RelationType;
```

Modify the following program code for getting the name of the related object of a unit by the RelatedObject attribute:

```
PlcSoftware plcTarget = GetControllerTargetByPLCName(Session.OpnsProject.Devices, PLCName);
PlcUnitProvider provider = plcTarget.GetService<PlcUnitProvider>();
//assuming existing units
m_PlcUnit = provider.UnitGroup.Units[0];
string unitRelatedObjectName = m_PlcUnit.Relations[1].RelatedObject;
```

Modify the following program code for identifying one relation in the collection using Find with the name of accessible element (RelatedObject):

```
PlcSoftware plcTarget = GetControllerTargetByPLCName(Session.OpnsProject.Devices, PLCName);
PlcUnitProvider provider = plcTarget.GetService<PlcUnitProvider>();
//assuming existing units
m_PlcUnit = provider.UnitGroup.Units[0];
PlcUnitRelation relation = m_PlcUnit.Relations.Find(unitRelatedObjectName);
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

7.19.25.5 Updating software unit properties

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open
See [Opening a project \(Page 106\)](#)

Application

You can use the TIA Portal Openness to update unit properties such as Author and Comment.

Program code: Update unit's author property

Modify the following program code to modify unit's author property.

```
//Set value
string newAuthor = "Z012345";
unit1.Author = newAuthor;
//Using SetAttributes()
string newAuthor = "Z012345";
var attrList = new List<KeyValuePair<string, object>>()
{
    new KeyValuePair<string, object>("Author", newAuthor)
};
unit1.SetAttributes(attrList);
//Using SetAttribute()
string newAuthor = "Z012345";
IEngineeringObject unit = (IEngineeringObject)unit2;
unit.SetAttribute("Author", newAuthor);
```

Error Message

Error will occur if you try to use the wrong value, for example starts with space, or contains illegal characters. An Recoverable Exception will be thrown with the following error message: "The value of attribute 'Author' contains an invalid character at Position 0."

```
PlcUnit unit1 = unitComposition.Create("Unit1");
try
{
    unit1.Author = " Author";
}
catch (EngineeringTargetException e)
{
    Console.WriteLine(e.Message);
}
```

Program code: Update unit's comment property

You cannot set or get the unit 'Comment' property directly as it is a MultilingualText. You can update the Comment Item's Text Property. The Comment.Item contains the cultures of the TIA Project. It can be adjusted via value assignment. The project languages are indexed in the Comment.Item composition, 0 marked as the first default project language. If there are more languages set, you can iterate on it, otherwise if the culture does not exist, an EngineeringTargetException is thrown.

```
//Setting the default first culture comment Item text:
unit1.Comment.Items[0].Text = "new Comment";
//Setting other culture comment Item Text:
unit1.Comment.Items[1].Text = "neuro Kommentar";
```

Error Message

Error will occur if you try to use the SetAttribute, SetAttributes, GetAttribute, GetAttributes. An EngineeringNotSupportedException will be thrown with the following error message: "Comment is not supported by type 'Siemens.Engineering.SW.Units.PlcUnit'."

```
try
{
    ((IEngineeringObject)unit1).SetAttribute("Comment", "new comments");
}
catch (EngineeringNotSupportedException e)
{
    Console.WriteLine(e.Message);
}
```

Error will occur if you try to refer a non existing culture. An EngineeringTargetException will be thrown with the following error message: "The argument 'index' (3) is outside the valid value range."

```
try
{
    unit1.Comment.Items[3].Text = "new Comment";
}
catch (EngineeringTargetException e)
{
    Console.WriteLine(e.Message);
}
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

7.19.25.6 Publishing software unit object

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)

Application

You can use the TIA Portal Openness to get or set the "Access" attribute of the underlying objects of units so that you can control their accessibility between units.

The following objects are supported in Openness units:

- Program blocks and data blocks - except OBs
- PLC types

The following possible value of the Access attribute represented as enumeration UnitAccessType:

- Published
- Unpublished

Note

The attribute 'Access' is only available on objects located under a unit, so if it is going to be set or get not under PlcUnit related objects a EngineeringNotSupportedException is thrown. In case of a wrong value (e.g.invalid enum values) EngineeringTargetInvocationException is thrown.

Program code: Configuring the Access attribute of PLC blocks

```
//Getting attribute value with GetAttribute
...
PlcBlock block = m_PlcUnit.BlockGroup.Find("FB_1");
UnitAccessType currentAccessValue = (UnitAccessType)block.GetAttribute("Access");
...
//Getting attribute value with GetAttributes
...
PlcBlock block = m_PlcUnit.BlockGroup.Block.Find("FB_1");
List<string> attrList = new List<string>();
{
    "Access"
};
IList<object> getAttributeValues = block.GetAttributes(attrList));
...
//Setting attribute value with SetAttribute
...
PlcBlock block = m_PlcUnit.BlockGroup.Blocks.Find("FB_1");
UnitAccessType newAccessValue = UnitAccessType.Published;
block.SetAttribute("Access", newAccessValue);
...
//Setting attribute value with SetAttributes
...
PlcBlock block = m_PlcUnit.BlockGroup.Blocks.Find("FB_1");
UnitAccessType newAccessValue = UnitAccessType.Published;
IList attrList = new List<KeyValuePair<string, object>>()
{
    new KeyValuePair<string, object>("Access", newAccessValue),
};
block.SetAttributes(attrList);
...
```

Program code: Configuring the Access attribute of PLC types

```

//Getting attribute value with GetAttribute
...
PlcType type = m_PlcUnit.TypeGroup.Types.Find("UDT_1");
UnitAccessType currentAccessValue = (UnitAccessType)type.GetAttribute("Access");

...
//Getting attribute value with GetAttributes
...

PlcType type = m_PlcUnit.TypeGroup.Types.Find("UDT_1");
List<string> attrList = new List<string>()
{
    "Access"
};
IList<object> getAttributeValues = type.GetAttributes(attrList);

...
//Setting attribute value with SetAttribute
...

PlcType type = m_PlcUnit.TypeGroup.Types.Find("UDT_1");
UnitAccessType newAccessValue = UnitAccessType.Published;
type.SetAttribute("Access", newAccessValue);

...
//Setting attribute value with SetAttributes
...

PlcType type = m_PlcUnit.TypeGroup.Types.Find("UDT_1");
UnitAccessType newAccessValue = UnitAccessType.Published;
IList attrList = new List<KeyValuePair<string, object>>()
{
    new KeyValuePair<string, object>("Access", newAccessValue),
};
type.SetAttributes(attrList);
...

```

See also

- Connecting to the TIA Portal (Page 76)
- Opening a project (Page 106)

7.19.25.7 Adding external sources in units

Requirement

- The TIA Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)

Application

You can use the TIA Portal Openness to add an external source under a software units. You can add source files such as .SCL,, DB, UDT under units.

You can perform the following tasks with external source group under software units while using Openness API :

- Adding ExternalSourceGroup under units
- Generating blocks/UDTs from an external source under software units
- Exporting source from blocks/UDTs under units

Program code: Adding an external source group

Modify the following program code to add an external source under a software unit:

```
PlcUnitProvider unitProvider = controllerTarget.GetService<PlcUnitProvider>();  
PlcUnit newUnit = unitProvider.UnitGroup.Units.Create("Unit_1");  
PlcExternalSource unitExternalSource =  
newUnit.ExternalSourceGroup.ExternalSources.CreateFromFile(externalSourceFilename,  
GetFilePath(externalSourceFilename));
```

Program code: Generating blocks/UDT from source

Modify the following program code to generate blocks\UDTs with void return types from an external source under a software unit:

```
unitExternalSource.GenerateBlocksFromSource();
```

Modify the following program code to generate blocks\UDT with collection of blocks get generated from an external source under a software unit:

```
// Blocks/UDTs gets generated only if there are no compilation errors  
IList<IEngineeringObject> generatedBlocksUnderUnitList =  
unitExternalSource.GenerateBlocksFromSource(GenerateBlockOption.None);  
// Blocks/UDTs will be generated regardless of any compilation errors  
IList<IEngineeringObject> generatedBlocksUnderUnitList =  
unitExternalSource.GenerateBlocksFromSource(GenerateBlockOption.KeepOnError);
```

Program code: Exporting source from blocks/UDTs

Modify the following program code to generate source from blocks under a software unit:

```
// For blocks
PlcBlock unitPlcBlock = newUnit.BlockGroup.Blocks.Find(generatedBlockName);
// Generate source from blocks with dependencies
newUnit.ExternalSourceGroup.GenerateSource(new[] { unitPlcBlock }, new
FileInfo(outputFileGeneratedPath), GenerateOptions.WithDependencies);
// Generate source from blocks without dependencies
newUnit.ExternalSourceGroup.GenerateSource(new[] { unitPlcBlock }, new
FileInfo(outputFileGeneratedPath), GenerateOptions.None);
```

Modify the following program code to generate source from UDTs under a software unit:

```
// For UDTs
PlcType unitPlcUdt = newUnit.TypeGroup.Types.Find(generatedUdtName);
//Generate source from UDTs with dependencies
newUnit.ExternalSourceGroup.GenerateSource(new[] { unitPlcUdt }, new
FileInfo(outputFileGeneratedPath), GenerateOptions.WithDependencies);
//Generate source from UDTs without dependencies
newUnit.ExternalSourceGroup.GenerateSource(new[] { unitPlcUdt }, new
FileInfo(outputFileGeneratedPath), GenerateOptions.None);
```

Program code: Enumerating external source file group

Modify the following program code to enumerate through the external source file group composition under the current unit:

```
PlcUnitProvider unitProvider = controllerTarget.GetService<PlcUnitProvider>();
PlcUnit newUnit = unitProvider.UnitGroup.Units.Find(textBoxAddNewUnit.Text);
PlcExternalSourceComposition unitExtSrcComposition =
newUnit.ExternalSourceGroup.ExternalSources;
foreach (PlcExternalSource unitExtSrc in unitExtSrcComposition)
{
    unitExtSrc.GenerateBlocksFromSource(GenerateBlockOption.None);
}
```

7.19.25.8 Units as mastercopies

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- A unit is created
See Working with software unit (Page 577)

Application

You can use the TIA Portal Openness to copy units as mastercopies to project library and global library.

You can perform the following possible tasks with unit as mastercopy while using TIA Portal Openness:

- Create unit as mastercopy in project library from software units
- Create unit as mastercopy in global library from software units
- Recreate unit from mastercopy of project library to software units in PNV.
- Recreate unit from mastercopy of global library to software units in PNV

Program code

Modify the following program code to create new unit as mastercopy in project library from software units using MasterCopyComposition:

```
PlcUnitProvider m_UnitProvider = plc.GetService<PlcUnitProvider>();
PlcUnitComposition m_UnitComposition = m_UnitProvider.UnitGroup.Units;
PlcUnit m_SoftwareUnit1 = unitComposition.Create("Unit_1");//Assuming existing units
IMasterCopySource m_UnitAsMasterCopy = (IMasterCopySource)m_SoftwareUnit1;//Assuming that
m_SoftwareUnit1 is present in the PLC
m_ProjectLibrary.MasterCopyFolder.MasterCopies.Create(m_UnitAsMasterCopy);
```

Modify the following program code to create unit as mastercopy in global library from software units:

```
...
IMasterCopySource m_UnitAsMasterCopy = (IMasterCopySource)m_SoftwareUnit1;//Assuming that
m_SoftwareUnit1 is present in the PLC
m_TiaPortal.GlobalLibraries.Open(libfile, OpenMode.ReadWrite);
GlobalLibrary m_GlobalLibrary = m_TiaPortal.GlobalLibraries[0];
m_GlobalLibrary.MasterCopyFolder.MasterCopies.Create(m_UnitAsMasterCopy)
...
```

Error will occur if you try to create a new unit in read-only global library. A recoverable exception will be thrown with the message "Cannot write to read-only libraries".

```
m_TiaPortal.GlobalLibraries.Open(libfile, OpenMode.ReadOnly);
GlobalLibrary m_GlobalLibrary = m_TiaPortal.GlobalLibraries[0];
try
{
m_GlobalLibrary.MasterCopyFolder.MasterCopies.Create(m_UnitAsMasterCopy);
}
catch (Exception e)
{
Console.WriteLine(e.Message);
}
```

Modify the following program code to recreate a unit from mastercopy of project library to software units in PNV:

```
...
ProjectLibrary m_ProjectLibrary = project.ProjectLibrary;
PlcUnitProvider m_UnitProvider = plc.GetService<PlcUnitProvider>();
PlcUnitComposition m_UnitComposition = m_UnitProvider.UnitGroup.Units;
...
MasterCopy mc_Unit_2 = m_ProjectLibrary.MasterCopyFolder.MasterCopies.Find("Unit_2");
m_UnitComposition.CreateFrom(mc_Unit_2); //Recreate a Unit from Project Library to
SoftwareUnits folder
```

Modify the following program code to recreate unit from mastercopy of global library to software units in PNV.

```
...
GlobalLibrary m_GlobalLibrary = m_TiaPortal.GlobalLibraries[0];
...
mc_Unit_2 = m_GlobalLibrary.MasterCopyFolder.MasterCopies.Find("Unit_2");
m_UnitComposition.CreateFrom(mc_Unit_2); //Recreate a unit from Global Library to
SoftwareUnits folder
```

See also

- [Connecting to the TIA Portal \(Page 76\)](#)
- [Opening a project \(Page 106\)](#)
- [Working with software unit \(Page 577\)](#)

7.19.25.9 Updating existing relations & create/delete relations

Requirement

- The TIA Openness application is connected to the TIA Portal
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open
See [Opening a project \(Page 106\)](#)

Application

You can use the TIA Portal Openness to modify the relations of a PLC units so that you can control the accessibility of objects existing in other PLC units or outside PLC units.

You can perform the following possible types of modification using TIA Portal Openness:

- Creating new relations
- Deleting existing relations
- Modifying existing relations

Program code: Creating new relations

You can create a new relations by providing the appropriate relation type and the name of the related object.

```
private PlcUnitRelation m_SoftwareUnitRelation;
private PlcUnitRelation m_NonUnitDBRelation;
private PlcUnitRelation m_TODBRelation;
...
PlcSoftware plcTarget = GetControllerTargetByPLCName(Session.OpnsProject.Devices, PLCName);
PlcUnitProvider plcUnitProvider = plcTarget.GetService<PlcUnitProvider>();
m_PlcUnit = plcUnitProvider.UnitGroup.Units[0]; //assuming existing units
```

Modify the following program code to create a new relation accessing object in another Plc units:

```
...
//assuming Plc unit "Unit_2" is already existing in the Plc
m_SoftwareUnitRelation = m_PlcUnit.Relations.Create("Unit_2",
UnitRelationType.SoftwareUnit);
...
```

Modify the following program code to create a new relation accessing PLC data block outside of Plc units:

```
...
//assuming Plc data block "Data_block_1" is already existing in the Plc
m_NonUnitDBRelation = m_PlcUnit.Relations.Create("Data_block_1",
UnitRelationType.NonUnitDB);
...
```

Modify the following program code to create a new relation accessing a Technological object outside of Plc units:

```
...
//assuming Technological object "SpeedAxis_1" is already existing in the Plc
m_TODBRelation = m_PlcUnit.Relations.Create("SpeedAxis_1", UnitRelationType.TODB);
...
```

Possible Error cases for creating a new relation:

Modify the following program code to create new relation accessing non existing related object:

```
...
//assuming Plc unit "NonExistingUnit" is not existing in the Plc yet
m_SoftwareUnitRelation = m_PlcUnit.Relations.Create("NonExistingUnit", UnitRelationType.
SoftwareUnit);
...
```

Note

In the above program code, the relation is created and no exception is thrown.

Modify the following program code to create a new relation by specifying a name for the related object which is not conform with the TIA naming rules:

```
...  
m_SoftwareUnitRelation = m_PlcUnit.Relations.Create(" Unit_2",  
UnitRelationType.SoftwareUnit);  
...
```

Note

In the above program code, the relation is not created and a recoverable exception is thrown due to leading spaces.

Modify the following program code to create a new relation from unit to itself:

```
...  
//assuming m_PlcUnit is assigned to Plc unit "Unit_1"  
m_SoftwareUnitRelation = m_PlcUnit.Relations.Create("Unit_1",  
UnitRelationType.SoftwareUnit);  
...
```

Modify the following program code to create a new relation which is duplicate of another already existing relation with the same relation type and related object:

```
...  
//assuming a relation with the same relation type and related object is already existing  
m_SoftwareUnitRelation = m_PlcUnit.Relations.Create("Unit_2",  
UnitRelationType.SoftwareUnit);  
...
```

Modify the following program code to create a new relation by specifying an empty string for the name of the related object:

```
...  
m_SoftwareUnitRelation = m_PlcUnit.Relations.Create(string.Empty,  
UnitRelationType.SoftwareUnit);  
...
```

Note

In all of the above program codes for error scenarios, the relation is not created and a recoverable exception is thrown.

Program code: Delete existing relations

You can create a new relations by providing the appropriate relation type and the name of the related object.

```
private PlcUnitRelation m_SoftwareUnitRelation;
private PlcUnitRelation m_NonUnitDBRelation;
private PlcUnitRelation m_TODBRelation;
...
PlcSoftware plcTarget = GetControllerTargetByPLCName(Session.OpnsProject.Devices, PLCName);
PlcUnitProvider plcUnitProvider = plcTarget.GetService<PlcUnitProvider>();
m_Plugin = plcUnitProvider.UnitGroup.Units[0]; //assuming existing units
m_Plugin2 = plcUnitProvider.UnitGroup.Units[1]; //assuming existing units
m_SoftwareUnitRelation = m_Plugin.Relations.Create("Unit_2",
UnitRelationType.SoftwareUnit);
m_NonUnitDBRelation = m_Plugin.Relations.Create("DB_Global", UnitRelationType.NonUnitDB);
m_TODBRelation = m_Plugin.Relations.Create("Axis_TO", UnitRelationType.TODB);
```

You can delete the relations in several ways.

Modify the following program code to delete the relation accessed by indexer:

```
...
m_Plugin.Relations[0].Delete();
...
```

Modify the following program code to delete the relation directly:

```
...
m_SoftwareUnitRelation.Delete();
...
```

Program code: Modify existing relations

You can modify the relations in several ways:

You can create a new relations by providing the appropriate relation type and the name of the related object.

```
private PlcUnitRelation m_Relation;
PlcSoftware plcTarget = GetControllerTargetByPLCName(Session.OpnsProject.Devices, PLCName);
PlcUnitProvider plcUnitProvider = plcTarget.GetService<PlcUnitProvider>();
m_Plugin = plcUnitProvider.UnitGroup.Units[0];
//assuming existing units
m_Plugin2 = plcUnitProvider.UnitGroup.Units[1];
//assuming existing units
m_Relation = m_Plugin.Relations.Create("Unit_2", UnitRelationType.SoftwareUnit);
...
```

Modify the following program code to update the name of the related object by assigning a new value for the RelatedObject attribute directly:

```
...
m_Relation.RelatedObject = "Unit_3";
...
```

Modify the following program code to update the name of the related object by assigning a new value for the RelatedObject attribute through SetAttribute:

```
...
m_Relation.SetAttribute("RelatedObject", "Unit_4");
...
```

Modify the following program code to update the name of the related object by assigning a new value for the RelatedObject attribute through SetAttributes:

```
...
IList attrList = new List<KeyValuePair<string, object>>();
{
new KeyValuePair<string, object>("RelatedObject", "Unit_4")
};
m_Relation.SetAttributes(attrList);
...
```

Possible error cases for modifying existing relations

Modify the following program code to update a relation to access a non existing related object:

```
...
//assuming Plc unit "NonExistingUnit" is not existing in the Plc yet
m_Relation.RelatedObject = "NonExistingUnit";
...
```

Note

In the above program code, the relation is modified and no exception is thrown.

Modify the following program code to update a relation by specifying a name for the related object which is not confirm with the TIA naming rules:

```
...
m_Relation.RelatedObject = " Unit_2";
...
```

Modify the following program code to update a relation to access itself:

```
...
//assuming m_Relation is defined under Plc unit "Unit_1"
m_Relation.RelatedObject = "Unit_1";
...
```

Note

In the above program code, the relation is not modified and a recoverable exception is thrown.

Modify the following program code to update a relation in a way which would be a duplicate of another already existing with the same relation type and related object:

```
...
//assuming a relation with the same relation type and related object is already existing
m_Relation.RelatedObject = "Unit_2";
...
```

Modify the following program code to update a relation by specifying an empty string for the name of the related object:

```
...
m_Relation.RelatedObject = string.Empty;
...
```

Note

In all of the above program codes for errors scenarios, the relation is not modified and a recoverable exception is thrown.

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.20 Functions for Version Control Interface

7.20.1 Accessing VCI system group in project

Requirement

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

You can use the TIA Portal Openness to navigate to the VCI system group from the project by accessing a `VersionControlInterface` from the project as a service. The `VersionControlInterface` itself is always a service and will return non-null object.

Program code

Modify the following program code to retrieve the workspace system group from the `VersionControlInterface`:

```
Siemens.Engineering.Project project = tiaPortal.Projects[0];
Siemens.Engineering.VersionControl.VersionControlInterface versionControlInterface =
project.GetService<VersionControlInterface>();
Siemens.Engineering.VersionControl.WorkspaceSystemGroup workspaceSystemGroup =
versionControlInterface.WorkspaceGroup;;
```

See also

[Opening a project \(Page 106\)](#)
[Connecting to the TIA Portal \(Page 76\)](#)

7.20.2 Enumerating user groups in VCI group

Requirement

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Program code

Modify the following program code to enumerate all VCI workspace user groups in other VCI workspace groups:

```
WorkspaceUserGroupComposition userGroupComposition = workspaceGroup.Groups;
foreach (Siemens.Engineering.VersionControl.WorkspaceUserGroup workspaceUserGroup in
userGroupComposition)
{
//...
}
```

Modify the following program code to access individual workspace user group in other VCI workspace groups:

```
Siemens.Engineering.VersionControl.WorkspaceUserGroup workspaceUserGroup =
workspaceGroup.Groups.Find("Some Group Name");
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.20.3 Creating VCI user group in VCI group

Requirement

- The Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a Project \(Page 106\)](#)

Application

You can use the TIA Portal Openness to create a workspace user group in a workspace group.

Program code

Modify the following program code to create a workspace user group:

```
VersionControlInterface versionControlInterface =
project.GetService<VersionControlInterface>();
WorkspaceSystemGroup workspaceGroupComposition =
versionControlInterface.WorkspaceGroup.Groups;
WorkspaceUserGroup result = workspaceGroupComposition.Create("NewWorkspaceUserGroup");
```

Note

To create a new workspace user group by name requires that supplied name should be valid and the supplied name does not already exist on another workspace user group

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.20.4 Updating VCI group properties

Requirement

- The Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)

Application

You can use the TIA Portal Openness to update the properties of workspace user group.

The following property is available to update on a workspace group:

Property Name	Return Type	Description	Accessibility
Name	System.String	The name of the workspace user group	Read/Write

Program code

Modify the following program code to update workspace user group name:

```
var workspaceUserGroup = ...;
workspaceUserGroup.Name = "New_Group_Name";
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.20.5 Deleting VCI user group

Requirement

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

You can use the TIA Portal Openness to delete workspace user groups. When you are deleting workspace user groups, all other objects contained in the workspace user group will also get deleted. This includes other workspace user groups and VCI workspaces.

Program code

Modify the following program code to delete a workspace user group:

```
WorkspaceUserGroup workspaceUserGroup = ...;  
workspaceUserGroup.Delete();
```

See also

Connecting to the TIA Portal (Page 76)

Opening a project (Page 106)

7.20.6 Enumerating VCI workspaces in VCI group

Requirement

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a Project (Page 106)

Program code

Modify the following program code to enumerate all VCI workspaces in a VCI group:

```
WorkspaceComposition workspaceComposition = workspaceSystemGroup.Workspaces;
foreach (Siemens.Engineering.VersionControl.Workspace workspace in workspaceComposition)
{
//...
}
```

Modify the following program code to find specific workspace by name:

```
Siemens.Engineering.VersionControl.Workspace workspace =
workspaceUserGroup.Workspaces.Find("SomeWorkspaceName");
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

7.20.7 Creating VCI workspace in VCI group

Requirement

- The Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a Project \(Page 106\)](#)

Application

You can use the TIA Portal Openness to create a workspace in a workspace user group.

You can create `Siemens.Engineering.VersionControl.Workspace` with the `create` action:

```
Siemens.Engineering.VersionControl.WorkspaceComposition.Create(string name)
```

Program code

Modify the following program code to create a workspace:

```
VersionControlInterface versionControlInterface =
project.GetService<VersionControlInterface>();
WorkspaceComposition workspaceComposition =
versionControlInterface.WorkspaceGroup.Workspaces;
Workspace result = workspaceComposition.Create("NewWorkspace");
```

Note

To create a new workspace by name requires that supplied name should be valid and the name does not already exist on another workspace user group.

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.20.8 Updating VCI workspace properties

Requirement

- The Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a Project \(Page 106\)](#)

Application

You can use the TIA Portal Openness to update the properties of a VCI workspace.

The following properties are available to update on a workspace:

Property name	Return type	Description	Accessibility
Name	System.String	The name of the workspace.	Read/Write
RootPath	System.IO.DirectoryInfo	The configured workspace path.	Read/Write

Program code: Updating name property

Modify the following program code to update the 'Name' property on workspace:

```
var workspace = ...;
workspace.Name = "New_Workspace_Name";
```

Program code: Updating rootpath property

Modify the following program code to configure the workspace path to a rooted directory info. Setting this value to null will unconfigure the workspaces root path.

```
var workspace = ...;
workspace.RootPath = new DirectoryInfo(@"D:\Project_WS");
```

Modify the following program code to unconfigure a workspace:

```
var workspace = ...;
workspace.RootPath = null;
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.20.9 Deleting VCI workspace

Requirement

- The Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a Project \(Page 106\)](#)

Application

You can use the TIA Portal Openness to delete a workspace. When a workspace is deleted all of the workspace's mappings get also deleted.

Program code

Modify the following program code to delete workspace:

```
Workspace workspace = ...;  
workspace.Delete();
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.20.10 Enumerating VCI workspace mappings in VCI

Requirement

- The Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)

Program code

Modify the following program code to enumerate all VCI workspace mappings in a workspace:

```
var workspace = ...;  
WorkspaceMappingComposition mappings = workspace.Mappings;  
foreach (Siemens.Engineering.VersionControl.WorkspaceMapping workspaceMapping in mappings)  
{  
//...  
}
```

Modify the following program code to access individual workspace mapping for the specific engineering object:

```
var workspace = ...;  
IEngineeringObject versionControlSupportedEngineeringObject = ...;  
Siemens.Engineering.VersionControl.WorkspaceMapping workspaceMapping =  
workspace.Mappings.Find(versionControlSupportedEngineeringObject);
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.20.11 Creating VCI workspace mapping in VCI workspace

Requirement

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

You can use the TIA Portal Openness to create a workspace mapping in VCI workspace.

You can create `Siemens.Engineering.VersionControl.WorkspaceMapping` with the `create` action signature:

```
Siemens.Engineering.VersionControl.WorkspaceMappingComposition.Create  
e(string relativeWorkspacePath, IEngineeringObject  
linkedProjectObject)
```

Program code

Modify the following code to create a workspace mapping:

```
var workspace = ...;  
var plcBlock = ...;  
var result = workspace.Mappings.Create(@"\TestCopy\Block_1.xml", plcBlock);
```

Recoverable Exception will be thrown in case of :

- The linked object is not a valid VCI supported object
- The linked object is already mapped.
- The relative file path contains invalid file characters.
- The relative file path contains parent directory navigation.

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.20.12 Updating VCI workspace mapping properties

Requirement

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

You can use the TIA Openness Portal to update the properties of a VCI workspace mapping.

The following properties are available on a workspace mapping.

Property name	Return type	Description	Accessibility
RelativeWorkspacePath	System.String	The relative file path to the linked objects representation in source control.	Read/write
LinkedProjectObject	IEngineeringObject	The engineering object compatible with VCI that is linked to the file in source control	Read

Program code

Modify the following program code to update the workspace mapping relative file path:

```
var workspaceMapping = ...;
workspaceMapping.RelativeWorkspacePath = @"\Test\NewBlock_1.xml";
```

See also

Connecting to the TIA Portal (Page 76)

Opening a project (Page 106)

7.20.13 Deleting VCI workspace mapping

Requirement

- The Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Principle

You can use the TIA Portal Openness to delete a VCI workspace mapping.

Program code

Modify the following program code to delete a VCI workspace mapping:

```
WorkspaceMapping workspaceMapping = ...;  
workspaceMapping.Delete();
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

7.20.14 Synchronizing a workspace mapping

Principle

- The Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)

Application

You can use the TIA Portal Openness to view the status of synchronization workspace mapping.

Program code: Individual Object Synchronization status

You can view status, update status and synchronize status of individual objects you must first request for

`Siemens.Engineering.VersionControl.WorkspaceMapping.IndividualObjectSynchronizationStatus` from workspace mapping as work

It is recommended to perform a null check before invoking the actions on individual object synchronization service as that mapping may or may not support individual object synchronization.

```
var individualObjectSynchronizationStatus =
workspaceMapping.GetService<IndividualObjectSynchronizationStatus>();
if(individualObjectSynchronizationStatus != null);
{
    //GetStatus()...
    //UpdateStatus()..
    //Synchronize()..
}
```

Program code: Get Status

You can view the synchronization status of individual object by using the action Siemens.Engineering.VersionControl.IndividualObjectSynchronizationStatus.GetStatus() which returns Siemens.Engineering.VersionControl.IndividualObjectCompareResult.

The IndividualObjectCompareResult object is used to inform about the current sync status.

```
public class IndividualObjectCompareResult
{
    CompareState CompareState { get; }
    IndividualObjectCompareDetails {get; }
}
```

The IndividualObjectCompareDetails flag enum is used to inform you about which part of the workspace mapping has changed (or both). This value will be None when the CompareState is Equal. If CompareState is unequal, this enum can have a value of ProjectObjectChanged or WorkspaceFileChanged or both ProjectObjectChanged, WorkspaceFileChanged.

```
[Flags]public enum IndividualObjectCompareDetails
{
    None = 0,
    ProjectObjectChanged = 1,
    WorkspaceFileChanged = 2
}
```

The CompareState enum is used to inform if an object has any changes.

```
public enum CompareState
{
    Equal,
    Unequal,
    WorkspaceFileMissing,
    Unknown
}
```

Modify the following program code to get the status of a workspace mapping:

```
var workspaceMapping = ...;
var individualObjectSynchronizationStatus =
workspaceMapping.GetService<IndividualObjectSynchronizationStatus>();
if(individualObjectSynchronizationStatus != null)
{
var compareResult = individualObjectSynchronizationStatus.GetStatus();
//when compareState is unequal
{
if(compareResult.CompareState == CompareState.Unequal)
{
if(compareResult.IndividualObjectCompareDetails ==
IndividualObjectCompareDetails.WorkspaceFileChanged)
{
//Workspace file has changed
}
elseif(compareResult.IndividualObjectCompareDetails ==
(IndividualObjectCompareDetails.ProjectObjectChanged |
IndividualObjectCompareDetails.WorkspaceFileChanged) )
{
//Both project object and workspace file has changed
}
}
}
}
```

Program code: Updating synchronization status

You can force the system to perform a live comparison between the currently linked project object and workspace file. This can result in a modification to sync status to reflect the current state of the system, and can be used to determine the status of a recently linked (but not synchronized) mapping without modifying either the existing project object or workspace file.

Modify the following program code to update the status of a workspace mapping:

```
var workspaceMapping = ...;
var individualObjectSynchronizationStatus =
workspaceMapping.GetService<IndividualObjectSynchronizationStatus>();
if(individualObjectSynchronizationStatus != Null)
{
individualObjectSynchronizationStatus.UpdateStatus();
}
```

Program code: Synchronizing workspace

You can force the system to perform a live comparison between the currently linked project object and workspace file. This can result in a modification to sync status to reflect the current state of the system, and can be used to determine the status of a recently linked (but not synchronized) mapping without modifying either the existing project object or workspace file.

7.21 Functions on OPC

The SynchronizationMode enum is used to inform the system which direction to perform the sync:

```
public enum SynchronizationMode
{
    ProjectToWorkspace,
    WorkspaceToProject
}
```

This action will attempt to sync the linked project object to/from the linked workspace file object.

Modify the following program code to synchronize a workspace:

```
var workspaceMapping = ...;
var individualObjectSynchronizationStatus =
workspaceMapping.GetService<IndividualObjectSynchronizationStatus>();
if(individualObjectSynchronizationStatus != null)
{
    individualObjectSynchronizationStatus.Synchronize(SynchronizationMode.ProjectToWorkspace);
}
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.21 Functions on OPC

7.21.1 Configuring OPC UA server secure communication protocol

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open
See [Opening a project \(Page 106\)](#)

Introduction

You can use the TIA Portal Openness application to configure OPC UA server with security policy "Basic256Sha256". The security policy Basic256Sha256 needs to be added to the Runtime Settings. RDP needs to compile the properties in the xml configuration file.

The defaults are Enabled, Sign, and Sign and Encrypt.

In the XML file <Project>\OPC\userver\OPCUaServerWinCCPro.xml, you need to set the security policy and security policies according to ES device configuration.

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <OPCUA_Server_WinCC xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ua="http://opcfoundation.org/UA">
3
4      <SecuredApplication>
5          <BaseAddresses>
6              <ua:String>opc.tcp://[HostName]:4861</ua:String>
7          </BaseAddresses>
8          <SecurityProfileUris>
9              <SecurityProfile>
10                 <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#None</ProfileUri>
11                 <Enabled>true</Enabled>
12             </SecurityProfile>
13             <SecurityProfile>
14                 <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic128Rsa15</ProfileUri>
15                 <Enabled>true</Enabled>
16             </SecurityProfile>
17             <SecurityProfile>
18                 <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256</ProfileUri>
19                 <Enabled>true</Enabled>
20             </SecurityProfile>
21             <SecurityProfile>
22                 <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256</ProfileUri>
23                 <Enabled>true</Enabled>
24             </SecurityProfile>
25         </SecurityProfileUris>
26     </SecuredApplication>
27
28     <ServerConfiguration>
29         <SecurityPolicies>
30             <SecurityPolicy>
31                 <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#None</ProfileUri>
32                 <MessageSecurityModes>None</MessageSecurityModes>
33             </SecurityPolicy>
34             <SecurityPolicy>
35                 <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic128Rsa15</ProfileUri>
36                 <MessageSecurityModes>Sign</MessageSecurityModes>
37             </SecurityPolicy>
38             <SecurityPolicy>
39                 <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic128Rsa15</ProfileUri>
40                 <MessageSecurityModes>SignAndEncrypt</MessageSecurityModes>
41             </SecurityPolicy>
42             <SecurityPolicy>
43                 <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256</ProfileUri>
44                 <MessageSecurityModes>Sign</MessageSecurityModes>
45             </SecurityPolicy>
46             <SecurityPolicy>
47                 <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256</ProfileUri>
48                 <MessageSecurityModes>SignAndEncrypt</MessageSecurityModes>
49             </SecurityPolicy>
50             <SecurityPolicy>
51                 <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256</ProfileUri>
52                 <MessageSecurityModes>Sign</MessageSecurityModes>
53             </SecurityPolicy>
54             <SecurityPolicy>
55                 <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256</ProfileUri>
56                 <MessageSecurityModes>SignAndEncrypt</MessageSecurityModes>
57             </SecurityPolicy>
58         </SecurityPolicies>

```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

7.21.2 Setting OPC UA security policy

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
Opening a project (Page 106)
- OPC UA server is activated

Application

You can use the TIA Portal Openness application to set the security policy in OPC UA. You can implement the security policy as a dynamic attribute of type flagged enum: `OpcUaSecurityPolicies`. The security policy is only available in TIA Portal Openness if the OPC UA server is activated. If the OPC UA server is deactivated, you will encounter `EngineeringNotSupportedException` while trying to access the security policy for any other unavailable attribute.

The below table shows the possible values can be found for security policies:

TIA UI name	Enum entry	Value	Remarks
-	<code>NoneSelected</code>	0	Is equivalent to TIA UI when no checkbox is selected.
No security	<code>OpcUaSecurityPolicies-None</code>	1	
Basic128Rsa15 - Sign	<code>OpcUaSecurityPolicies128RSAS</code>	2	
Basic128Rsa15 - Sign & Encrypt	<code>OpcUaSecurityPolicies128RSASE</code>	4	
Basic256 - Sign	<code>OpcUaSecurityPolicies256S</code>	8	
Basic256 - Sign & Encrypt	<code>OpcUaSecurityPolicies256SE</code>	16	
Basic256Sha256 - Sign	<code>OpcUaSecurityPolicies256SHAS</code>	32	
Basic256Sha256 - Sign & Encrypt	<code>OpcUaSecurityPolicies256SHASE</code>	64	

Program code

Modify the following program code to set the security policy in OPC UA using TIA Portal Openness:

```
DeviceItem UpcUaSubmodule= ...;
object SecurityPolicies = UpcUaSubmodule.GetAttribute("OpcUaSecurityPolicies");
if(SecurityPolicies | OpcUaSecurityPolicies.OpcUaSecurityPolicies256S ==
OpcUaSecurityPolicies.OpcUaSecurityPolicies256S)
{
//Do something
}
UpcUaSubmodule.SetAttribute("OpcUaSecurityPolicies",
OpcUaSecurityPolicies.OpcUaSecurityPolicies256S |
OpcUaSecurityPolicies.OpcUaSecurityPolicies256SHASE);
```

See also

- [Connecting to the TIA Portal \(Page 76\)](#)
- [Opening a project \(Page 106\)](#)

7.22 SiVArc Openness

7.22.1 Introduction

Introduction

TIA portal openness application allows you to instantiate SiVArc. You must need a client application to access TIA portal, and through openness feature launch SiVArc services. For more details on set up, and accessing Openness, see TIA portal user guide.

Setting up the application

To set up a client application, perform the following steps:

1. Create a console application. Add reference of Public API (Siemens.Engineering.dll) available at _deployed\TIAPV15SP1_11010001\PublicAPI\V15.1\ 936 Siemens.Engineerin.dll or from installed binaries location PublicAPI\V15.1\ 937 Siemens.Engineerin.dll
2. Add configuration details to the configuration file. For detailed information of configuration details and steps to access the public API, see TIA openness wiki.
3. To access Sivarc service, use the below mentioned API:

```
using (TiaPortal tia = new
TiaPortal(TiaPortaMode.WithUserInterface))
{
    Project myProject = tia.Projects.Open(new FileInfo(@"C:\Users
\z003exve\Documents\Automation\Project_Demo\Project_Demo.ap15"));
    //if SiVArc is not installed, user will not be able to access
    SiVArc service (compiler error)
    SiVArc sivarc =myproject?.GetService<SiVArc>();
    if (sivarc !=null)
    {
    }
}
}
```

7.22.2 SiVArc service properties

SiVArc service properties

The table below lists the supported properties and methods for SiVArc:

Property Name	Description	Data Type
AlarmRules	Anchor object for all alarm rule objects	AlarmRulesBrowsable
ScreenRules	Anchor object for all screen rule objects	ScreenRulesBrowsable
TextlistRules	Anchor object for all textlist rule objects	TextlistRulesBrowsable
TagRules	Anchor object for all tag rule objects	TagRulesBrowsable
CopyRules	Anchor object for all copy rule objects	CopyRulesBrowsable
Alarm Rules	Enumerate of all immediate first level alarm rules	AlarmRuleComposition
Groups	Enumerate of all immediate first level alarm rule groups	AlarmRuleGroupComposition
ScreenRules	Enumerate of all immediate first level screen rules	ScreenRuleComposition
ScreenRulesGroups	Enumerate of all immediate first level screen rule groups	ScreenRuleGroupComposition
TextlistRules	Enumerate of all immediate first level textlist rules	TextlistRuleComposition

Property Name	Description	Data Type
TextlistGroups	Enumerate of all immediate first level textlist rule groups	TextlistRuleGroupComposition
TagRules	Enumerate of all immediate first level tag rules	TagRuleComposition
TagRulesGroups	Enumerate of all immediate first level tag rule groups	TagRuleGroupComposition
CopyRules	Enumerate of all immediate first level copy rules	CopyRuleComposition
CopyRulesGroups	Enumerate of all immediate first level copy rule groups	CopyRuleGroupComposition

Following table lists the AlarmRule and AlarmRuleGroup composition. The same applies to other SiVArc objects.

Method Name	Parameter	Description	Data Type
Find	String- alarm rule /rule-group name	Finds the alarm rule/ alarm rule group from alarm rule/alarm group collection	AlarmRule
CreateFrom	MasterCopy – alarm rule/alarm rule group master copy	Copy alarm rule/alarm rule group master copy from library to project with default replace option	AlarmRule
CreateFrom	MasterCopy – alarm rule/alarm rule group master copy, CreateOptions- Rename/Replace	Copy alarm rule /alarm rule group master copy from library to project with create option	AlarmRule

7.22.3 Copying rules or groups from library

Requirement

- Launch TIA portal openness application. For more information on connections, see TIA portal user guide.
- An existing TIA portal project containing screen rule editor, screen rules group and master copy.

Case 1: When you copy rules/rule groups from master copy to screen rules editor

"The CreateFrom" global library allows rules and rule groups to be copied from the global library to the SiVArc rule editor. If successful, the API function will return ScreenRule/ ScreenRuleGroup. The following code explains how the rules or rule groups are copied from the master copy to the SiVArc editor:

```
// Finds screen rule master copy "Screen rule_1"
MasterCopy screenRuleMasterCopy =
    myProject.ProjectLibrary.MasterCopyFolder.MasterCopies.Find("Screen rule_1");

if (screenRuleMasterCopy != null)
{
    var rule = sivarc.ScreenRules.Rules.CreateFrom(screenRuleMasterCopy);
    if (rule != null)
    {
        Console.WriteLine("Copied Screen Rule Name: " + rule.Name);
        Console.WriteLine("Copied Screen Rule Comment: " + rule.Comment);
        Console.WriteLine("Copied Screen Rule Enabled: " + rule.Enabled);
        Console.WriteLine("Copied Screen Rule Condition: " + rule.Condition);
    }
}
```

By default, the behaviour will be replaced.

Case 2: When you copyrules/rule groups from master copy, the existing rules/rule groups can be renamed or replaced based on the second parameter input

If rules/rule groups in the master copy are already existing in the SiVArc editor, and if you are trying to copy them, the rules/rule groups gets renamed. The "CreateOptions" API will create the rules/rule groups in the SiVArc editor if they are not existing, else they replace the existing rules/rule groups. If successful, the API function will rename ScreenRule/ScreenRuleGroup. The following code snippet shows the replacing of rules/rule groups:

```
// Finds screen rule master copy "Screen rule_1"
MasterCopy screenRuleMasterCopy =
    myProject.ProjectLibrary.MasterCopyFolder.MasterCopies.Find("Screen rule_1");

if (screenRuleMasterCopy != null)
{
    var rule = sivarc.ScreenRules.Rules.CreateFrom(screenRuleMasterCopy, CreateOptions.Rename);
    if (rule != null)
    {
        Console.WriteLine("Copied Screen Rule Name: " + rule.Name);
        Console.WriteLine("Copied Screen Rule Comment: " + rule.Comment);
        Console.WriteLine("Copied Screen Rule Enabled: " + rule.Enabled);
        Console.WriteLine("Copied Screen Rule Condition: " + rule.Condition);
    }
}
```

7.22.4 Finding a screen rule and screen rule group

Requirement

- TIA portal openness application connected to TIA portal. For more information on connections, see TIA portal user guide.
- TIA portal project consisting of screen rules and screen rule groups.

Finding screen rules within screen groups

Case 1: Finding screen rules within screen rules editor.

The API `sivarcs.ScreenRules.Rules` allows you to find the available rules within the SiVArc screen rule editor as shown below:

```
// Collection of all immediate first level screen rules
ScreenRuleComposition screenRules = sivarcs.ScreenRules.Rules;
if(screenRules != null && screenRules.Count > 0)
{
    // Finds screen rule
    ScreenRule rule = screenRules.Find("Screen rule_7");
    if(rule != null)
    {
        Console.WriteLine("Screen Rule Name: " + rule.Name);
        Console.WriteLine("Screen Rule Comment: " + rule.Comment);
        Console.WriteLine("Screen Rule Enabled: " + rule.Enabled);
        Console.WriteLine("Screen Rule Condition: " + rule.Condition);
```

Case 2: Finding screen rule groups within screen rules editor.

The API `sivarcs.ScreenRule.Groups` allows you to find the available rules and rule groups within the SiVArc screen rule editor as shown below:

```

var groups = sivarc.ScreenRules.Groups;
if (groups != null && groups.Count > 0)
{
    var rule = groups.Find("Screen rule group").Rules.Find("Screen rule_2");
    if (rule != null)
    {
        Console.WriteLine("Found Screen Rule Name: " + rule.Name);
        Console.WriteLine("Found Screen Rule Comment: " + rule.Comment);
        Console.WriteLine("Found Screen Rule Enabled: " + rule.Enabled);
        Console.WriteLine("Found Screen Rule Condition: " + rule.Condition);
    }

    var group = groups.Find("Screen rule group").Groups.Find("Screen rule group_2");
    if (group != null)
    {
        Console.WriteLine("Found Screen Rule Group Name: " + group.Name);
        Console.WriteLine("Found Screen Rule Group Comment: " + group.Comment);
        Console.WriteLine("Found Screen Rule Group Enabled: " + group.Enabled);
        Console.WriteLine("Found Screen Rule Group Condition: " + group.Condition);
    }
}

```

7.22.5 Deleting rules and rule groups

Requirement

- TIA portal openness application connected to TIA portal. For more information on connections, see TIA portal user guide.
- TIA project containing screen rules and screen rule groups.

Deleting rules and rule groups

To delete a rule or a rule group, the following API is used:

```
sivarc.ScreenRules.Rules.Find("Screen rule_1").Delete();
```

The ScreenRules is an anchor object for all screenrule objects. To perform deletion, it is mandatory that you find the rule within the rule editor. For more information on finding the screen rule, refer section Finding a screen rule and screen rule group.

To delete screens from screen group, use the following API:

```
sivarc.ScreenRules.Rules.Find("Screen rule group_1").Delete();
```

7.22.6 UMAC set up for openness

About UMAC

To access UMAC through openness, ensure that you have the UMAC credentials and access privileges. If you are not a valid UMAC user, the application will return `NULL` values for all SiVArc anchor rule objects. For more information on UMAC, refer UMAC topic in SiVArc user guide.

7.22.7 SiVArc generation

Requirement

- Launch TIA portal openness application. For more information on connections, see TIA portal user guide.
- An existing TIA portal project connected to an HMI device, and PLC configured.

Important points to notice

- Ensure SiVArc license is installed on your PC, else during generation an exception is thrown - "*SiVArc license is missing; it is mandatory to have SiVArc license for modifying data*".
- Ensure valid device name is used, else an exception is thrown - "*HMI device 'deviceName' not found*".
- Ensure valid PLC name is called, else an exception is thrown - "*PLC device 'plcDeviceName' not found*".
- Ensure supported device name is called, else an exception is thrown - "*HMI device 'deviceName' is not supported*"
- Ensure supported PLC name is called, else an exception is thrown - "*PLC device 'plcDeviceName' is not supported*"
- Ensure you pass valid GenerationOption parameter. If none is passed, SiVArc generation will happen and by default the TIAP project settings will be used for SiVArc generation
- Ensure you valid PLC name which was not used for generation in the previous generation, else the system freezes.

GenerationOptions- Enum (Flag)

SiVArc supports Enum options, and you can pass combination of two values in the Generate API. Following table displays the enum options:

SN	Values	Description
1	None	Nothing is selected, takes default setting for generation
2	AllTags	Generate all tags
3	UsedHmi-Tags	Generate only relevant (used) tags
4	FullGeneration	In case when FullGeneration option is not selected, SiVArc will decide internally based on the configuration if it has to perform full generation or delta generation. When you pass <code>FullGeneration</code> as parameter, SiVArc will be generated with force full generation.

To generate SiVArc, use the following API:

```
sivarc.Generate("HMI_1", new List<string> {PLC_1},
GenerateOptions.AllTags | GenerateOptions.FullGeneration);
```

SiVArcGenerationResult and SivarcFeedbackMessage

SiVArc generation accesses the following properties on successful generation:

- `IsGenerationSuccessful` - Informs if SiVArc generation is successful.
- `WarningCount` - Total warning count after SiVArc generation
- `ErrorCount` - Total error count after SiVArc generation
- `Messages` - Composition of feedback message

To generate SiVArc result, use the following API:

- Print Sivarc generated feedback messages

```
private void WriteSivarcGenerationResults(SivarcGenerationResult result)
{
    sb.Append("Is SiVArc generation successful:" +
    result.IsGenerationSuccessful);
    sb.Append(Environment.NewLine);
    sb.Append("Total Warning Count:" + result.WarningCount);
    sb.Append(Environment.NewLine);
    sb.Append("Total Error Count:" + result.ErrorCount);
    sb.Append(Environment.NewLine);

    RecursivelyWriteMessages(result.Messages);
}
```

SiVArc generation accesses the following feedback messages on successful generation:

- Path: Header text of feedback message(Header messages will always have empty description field)
- DateTime: DateTime of feedback message
- MessageType: Feedback message type
- Description: Description/Content of Feedback message (Only when Path is empty to ensure not a header message)
- WarningCount: Warning count for a header message
- ErrorCount: Error count for a header message
- Messages: Composition of feedback message (SivarcFeedbackMessage)

You can view recursive feedback messages by using the following code snippet:

```
private void RecursivelyWriteMessages(SivarcFeedbackMessageComposition messages)
{
    foreach (SivarcFeedbackMessage message in messages)
    {
        sb.Append("Path: " + message.Path);
        sb.Append(Environment.NewLine);
        sb.Append("DateTime: " + message.DateTime);
        sb.Append(Environment.NewLine);
        sb.Append("State: " + message.MessageType);
        sb.Append(Environment.NewLine);
        sb.Append("Description: " + message.Description);
        sb.Append(Environment.NewLine);
        sb.Append("Warning Count: " + message.WarningCount);
        sb.Append(Environment.NewLine);
        sb.Append("Error Count: " + message.ErrorCount);
        sb.Append(Environment.NewLine);
        sb.Append(Environment.NewLine);

        RecursivelyWriteMessages(message.Messages, indent);
    }
}
```

7.23 Openness for CP 1604/CP 1616/CP 1626

General

You can use the TIA Portal Openness application to configure transfer areas and transfer area mapping rules for the communication processors CP 1604/CP 1616 as of V2.8 (also as of V2.7 depending on the article number) and CP 1626 as of V1.1.

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
See "Establishing a connection to the TIA Portal".
- A project is open.
See "Open project".
- To compile the project, all devices must be "offline".

Configuration of transfer areas

Creating transfer areas

For example, to create a "CD" type transfer area for a CP 1604, use the following program code:

```
NetworkInterface cpItf = CP
1604Interface.GetService<NetworkInterface>();
// Create TransferAreas
TransferAreaComposition transferAreas = cpItf.TransferAreas;

// Simple TranferArea of type Input
TransferArea transferAreaInput =
    transferAreas.Create("Input CD", TransferAreaType.CD);
```

Attribute	Description	
name	Specifies the name of the transfer area to be created.	
type	Specifies the type of the transfer area to be created. The following types are possible:	
	TransferAreaType.CD	Data exchange controller device
	TransferAreaType.F_PS	Data exchange PROFIsafe
	TransferAreaType.TM	Transfer module mapping
Note: It is not possible to change the type at a later date.		

Setting attributes of the transfer areas

To set attributes of a transfer area, use the following program code, for example:

```
transferAreaTm.LocalToPartnerLength = 8;
transferAreaTm.Direction = TransferAreaDirection.LocalToPartner;
string name = transferAreaTm.Name
```

Some attributes must be set or queried, but all of them can be set or queried using the general calls "GetAttribute()" or "SetAttribute()". Use the following program code, for example:

```
const string myIndividualComment = "MyIndividualComment";
transferAreaTm.SetAttribute("Comment", myIndividualComment);
```

```
Int32 updateTime = transferAreaTm.GetAttribute("TransferUpdateTime")
```

Attribute	Description	
Name (string)	Specifies the name of the transfer area.	
Direction	Specifies the direction in which the data of the transfer area is transferred. The following directions are possible:	
	TransferAreaDirection.LocalToPartner	Data of the transfer area is transferred from the IO device to the higher-level IO controller.
	TransferAreaDirection.partnerToLocal	Data of the transfer area is transferred from the higher-level IO controller to the IO device.
	TransferAreaDirection.bidirectional	Data of the transfer area can be transferred in both directions between the higher-level IO controller and the IO device. The "LocalToPartnerLength" attribute determines the length of the transferred data from IO device to the higher-level IO controller. The "PartnerToLocalLength" attribute determines the amount of data from the higher-level IO controller to the IO device
Comment (string)	Text box for a comment on the transfer area.	
LocalToPartnerLength	Specifies the data length of the transfer area that is transferred from the IO device to the higher-level IO controller.	
PartnerToLocalLength	Specifies the data length of the transfer area that is transferred from the higher-level IO controller to the IO device.	
LocalAdresses	Specifies the input and output addresses of the transfer area from the local device.	
PartnerAdresses	Specifies the input and output addresses of the transfer area in the higher-level IO controller.	
TransferUpdateTime(Int32)	Specifies the update time of the transfer area. Only set or queried for a transfer area of the type "TransferAreaType.TM".	
PositionNumber	Specifies the number of the virtual submodule of this transfer area.	
Type	Specifies the type of transfer area, read-only.	
TransferAreaMappingRules	Specifies the routing table of the routing area, read-only.	

Deleting transfer areas

To delete transfer areas, use the following program code:

```
transferAreaInput.Delete();
```

Iteration through transfer areas

To iterate transfer areas, use the following program code:

```
TransferAreaComposition transferAreas = cpItf.TransferAreas;
foreach (TransferArea transferArea in transferAreas)
{
    transferArea.Delete();
}
```

Configuration of IO routing

Creating IO routes

To create IO routes, use the following program code:

```
// Create TransferAreaMappingRule
TransferAreaMappingRuleComposition routingTable
    = transferArea.TransferAreaMappingRules;

// Create a new IO route
TransferAreaMappingRule route1 =
    routingTable.Create();
```

Setting attributes of IO routes

The following attributes can be set for IO routing:

Attribute	Description
Offset	Bit-based offset within the routing area to which the data is to be assigned. The length of the offset is determined by the "Begin" and "End" attributes.
Target	Specifies the module or submodule of the IO device that contains the data to be assigned to the configuration of the IO device, a transfer area of the type "TM".
IoType	The "IoType" attribute can only be changed in a transfer area of the "Input" type. In addition, a mixed module must be configured as "Target" for this transfer area. Only then can you select whether the data of the inputs (IoType.Input) are to be read or whether the data of the outputs (IoType.Output) are to be read (back).
Begin	Specifies the beginning of the data to be read by the "Target" attribute.
End	Specifies the end of the data to be read by the "Target" attribute.

Deleting IO routes

To delete IO routes, use the following program code:

```
transferAreaMappingRule.Delete();
```

Iteration via IO routing

To iterate via IO routing, use the following program code:

```
TransferAreaMappingRuleComposition routingTable =
    transferArea.TransferAreaMappingRules;

foreach (TransferAreaMappingRule route in routingTable)
{
    route.Delete();
}
```

7.24 Openness transfer areas for PnPn coupler

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal
- A project is open
See Opening a project
- To compile the project, all device must be "offline"

Introduction

You can use the TIA Portal Openness to add module/submodule, delete and search transfer areas for PnPn coupler.

Configuration of transfer areas

Creating transfer areas

For example, to create module on the next free position, use the following program code:

```
NetworkInterface coupler_Itf = deviceItem.GetService<NetworkInterface>();  
TransferAreaComposition transferAreas = coupler_Itf.TransferAreas;  
// Create a transfer area of type input at next free positionnumber  
TransferArea transferAreaInput = transferAreas.Create("Input", TransferAreaType.IN);
```

Note

It is not possible to create a submodule without Positionnumber.

For example, to create module on PositionNumber and submodule at the next free submodulepositionnumber at positionNumber, use the following program code:

```
// Create a transfer area of type input at positionnumber 5  
TransferArea transferAreaInput = transferAreas.Create("Input", TransferAreaType.IN, 5);
```

7.24 Openness transfer areas for PnPn coupler

Attribute	Description	
Type	Specifies the type of the transfer area to be created. The following types are possible:	
	TransferAreaType.None	Default value
	TransferAreaType.IN	Transfer area Input
	TransferAreaType.OUT	Transfer area Output
	TransferAreaType.MSI	Transfer area Shared Input (MSI)
	TransferAreaType.MSO	Transfer area Shared Output (MSO)
	TransferAreaType.MSO_LOCAL	Transfer area Local Shared Output (MSO_LOCAL)
	TransferAreaType.RE-CORD_WRITE_STO	Transfer area for Record Data Write up to 8 data records will be buffered
	TransferAreaType.RE-CORD_WRITE_PUB	Transfer area for Record Data Write overwrites previous data record
	TransferAreaType.RE-CORD_READ_STO	Transfer area for Record Data Read reads the oldest data record in the buffer
	TransferAreaType.RE-CORD_READ_PUB	Transfer area for Record Data Read reads data record written last
	TransferAreaType.MSI_MSO	Transfer area Shared Input (MSI) / Transfer area Shared Output (MSO)
	TransferAreaType.IN_OUT	Transfer area Input / Transfer area Output
	TransferAreaType.LOCAL_RE-CORD_STO	Transfer area for local data record coupling, up to 8 data records are buffered
	TransferAreaType.LOCAL_RE-CORD_PUB	Transfer area for local data record coupling, overwrites previous data record
	TransferAreaType.SUB_MSI	Transfer area copy of Shared Input (MSI)
	TransferAreaType.SUB_MSO	Transfer area copy of Shared Output (MSO)
	TransferAreaType.SUB_LOCAL_RE-CORD_STO_READ	Transfer area for Record Data Read for reading the oldest data record in the buffer
	TransferAreaType.SUB_LOCAL_RE-CORD_PUB_READ	Transfer area for Record Data Read for reading the most recently written data record
	TransferAreaType.PROFI-SAFE_IN12_OUT6	Transfer area with 12 byte input and 6 byte output
	TransferAreaType.PROFI-SAFE_IN6_OUT12	Transfer area with 6 byte input and 12 byte output

Setting attributes of the transfer areas

To set attributes of a transfer area, use the following program code, for example:

```
// Change input length
CouplerTransf.LocalToPartnerLength = 5;
// Change output length
CouplerTransf.PartnerToLocalLength = 5;
// Change name
CouplerTransf.Name = "Testname";
```

Some attributes must be set or queried, but all of them can be set or queried using the general calls TransferArea.GetAttribute() or SetAttribute(). Use the following program code, for example

```
// Set Comment of Transferarea
string myIndividualComment = "MyIndividualComment";
CouplerTransf.SetAttribute("Comment", myIndividualComment);
// Get positionNumber of Transferarea
Int32 positionNumber = (Int32)CouplerTransf.GetAttribute("PositionNumber");
```

Attribute	Description	
Name (string)	Specifies the name of the transfer area	
Direction	Specifies the direction in which the data of the transfer area is transferred. The following directions are possible:	
	TransferAreaDirection.localTo- Partner	Data of the transfer area is transferred from the IO device to the higher-level IO controller.
	TransferAreaDirection.partnerTo- Local	Data of the transfer area is transferred from the higherlevel IO controller to the IO device.
	TransferAreaDirection.bi-directional	Data of the transfer area can be transferred in both directions between the higher-level IO controller and the IO device. The "LocalToPartnerLength" attribute determines the length of the transferred data from IO device to the higher-level IO controller. The "PartnerToLocal- Length" attribute determines the amount of data from the higher-level IO controller to the IO device
Comment (string)	Text box for a comment on the transfer area	
LocalToPartnerLength	Specifies the input data length of the transfer area	
PartnerToLocalLength	Specifies the output data length of the transfer area	
PositionNumber	Specifies the slot number of the transfer area	
ExtendedPositionNumber	Specifies the sub slot number of the transfer area	
Type	Specifies the type of transfer area	
SharedDeviceAccessConfigured	Controls the access to PLC	
UpdateAlarm	Enable the update alarm of the transfer area	
RecordIndex	Set the data record number that you will be using when writing the data record	

Deleting transfer areas

To delete transfer areas, use the following program code:

```
TransferArea TransferAreaInput = transferAreas.Create("Input", TransferAreaType.IN);  
TransferAreaInput.Delete();
```

Searching transfer areas

To get the TransferArea with positionNumber and extendedPositionNumber, use the following program code:

```
// Find a transfer area with Positionnumber and ExtendedPositionNumber  
TransferArea transferAreaFind = transferAreas.Find(4, 3);
```

Note

ExtendedPositionNumber is only needed for TransferAreas with more than one Submodule like MSI

To get the first TransferArea with this positionNumber, use the following program code:

```
// Find a transfer area with Positionnumber  
TransferArea transferAreaFind = transferAreas.Find(5);
```

Note

If one of this attributes is not available, you will encounter an exception.

7.25 Openness virtual modules/submodules for ET 200SP PN HF

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal
- A project is open
See Opening a project

Introduction

You can use the TIA Portal Openness to add and delete virtual modules/submodule for ET 200 SP PN HF.

The following property are supported in virtual modules:

Property name	Data type	EomAtomValue	EomAtom description
SharedDeviceAccessConfigured	Boolean		
Name	String		
PositionNumber	Int32		
VirtualType	UInt64	1	MsoLocal
AddSubModules	Int64		

The following property are supported in virtual submodules:

Property	Data type	EomAtom value	EomAtom Description
SharedDeviceAccessConfigured	Boolean		
Comment	String		
PositionNumber	Int32		
Name	String		
ActivateDataStatus	Boolean		
InputAddressLength	Int64		
OutputAddressLength	Int64		
VirtualSubType	UInt64	1	MsoLocal
		2	MsoSub

Program code: Create and delete virtual module

Modify the following program code to add a new module:

```
string Type = "OrderNumber:6ES7 155-6AU30-0CN0/V4.2";
Device ET200SP = newProject.Devices.CreateWithItem(Type, "ET200SP",
"ET200SP");
DeviceItem Rack = ET200SP.DeviceItems.First();
string TypeIdentifier = "OrderNumber:ET200SP.Virtual.Module";
string Name = "VirtualIO_1";
int PositionNumber = 100;
DeviceItem VIM = Rack.PlugNew(TypeIdentifier, Name, PositionNumber);
```

Modify the following program delete a module

```
DeviceItem Rack = ET200SP.DeviceItems.First();
string TypeIdentifier = "OrderNumber:ET200SP.Virtual.Module";
string Name = "VirtualIO_1";
int PositionNumber = 100;
DeviceItem VIM = Rack.PlugNew(TypeIdentifier, Name, PositionNumber);
VIM.Delete();
```

Program code: Create and delete virtual sub module

Modify the following program code to create virtual submodules in Openness:

```
VIM1.SetAttribute ("AddSubModules", (Int64)1);  
VIM2.SetAttribute ("AddSubModules", (Int64)2);
```

Note

To add a submodule, you have to set the module property "AddSubModules" to count of added submodules. If you try to add more submodules than allowed, there is an exception.

Modify the following program code to delete a submodule:

```
SubModule3.Delete();
```

Note

You cannot delete the first two submodules.

7.26 Functions for SINUMERIK

7.26.1 Introduction

Using TIA Portal-Openness, you automate the engineering and control the TIA Portal using a program that you created yourself.

In this help document, you can find a lot of information and code examples for this program that you generate yourself. You can also generate and use your own programs for the TIA Portal "SINUMERIK" application.

Further information

Before you generate your own program for SINUMERIK from the sample codes listed in the following, please note the general information on Openness, which you can find in this help under the following keywords:

- Preconditions for using TIA Portal Openness
- Installing TIA Portal Openness
- Access the TIA Portal
- TIA Portal Openness object model
- Programming steps

7.26.2 Type identifier - identifier of the components

Each SINUMERIK component has a unique number, which is designated as type identifier (TypeIdentifier). In the Openness program code, you can use the type identifier to clearly identify and designate a component. For example, the type identifier for SINUMERIK 840D sl NCU 710.3 PN is "Article No. :6FC5 371-0AA30-0Axx/Vy.z".

The type identifier is displayed when creating a device using the dialog "Add new device" and in the tabular device overview.

You can copy the type identifier into your Openness application.

The type identifier is expected in Openness as a current parameter when a method is called, e.g. the CreateWithItem() method.

Activating display of the type identifier in SINUMERIK

1. In the project view, choose "Options > Settings" from the menu.
The "Settings" configuration area opens.
2. In the secondary navigation, select entry "Hardware configuration".
3. Activate option "Activate display of the type identifier for devices and modules".
The type identifier display is now active.

7.26.3 Fundamentals

Overview

You can create the following SINUMERIK devices in the TIA Portal Openness:

- NCU
- NX module
- ADI4

To create the SINUMERIK devices, use the `CreateWithItem()` method of the `Devices` Collection.

Note

All integrated subcomponents of a SINUMERIK-NCU - such as PLC, NCK, CP, HMI and SINAMICS Integrated - are automatically created at the same subordinate level.

Special issue when creating SINUMERIK devices

The names of the subracks correspond to the NCU types and are write protected in the user interface. You must also use the names of the subracks in the TIA Portal Openness.

Use the following standard names of the subracks for creating a device using a device element type identifier with the `Device CreateWithItem` method:

SINUMERIK NCU	Subrack standard name
SINUMERIK 840D sl NCU 710.3 PN	NCU 710.3 PN
SINUMERIK 840D sl NCU 720.3 PN	NCU 720.3 PN
SINUMERIK 840D sl NCU 730.3 PN	NCU 730.3 PN
SINUMERIK 840D sl NCU 730.3 PN /319	NCU 730.3 PN /319

Note

Alternatively, you can omit the parameter names. The default name is used if the name is "Null" or "String.Empty".

The following parameters can be used with the `CreateWithItem()` method:

- default name,

e.g.

```
project.Devices.CreateWithItem("OrderNumber:6FC5 371-0AA30-0AAB/
V4.8", "NCU 710.3 PN", "TestDevice");
```

- null,

e.g.

```
project.Devices.CreateWithItem("OrderNumber:6FC5 371-0AA30-0AAB/
V4.8", null, "TestDevice");
```

- `string.Empty`,

e.g.

```
project.Devices.CreateWithItem("OrderNumber:6FC5 371-0AA30-0AAB/
V4.8", string.Empty, "TestDevice");
```

Additional information about the call parameters associated with the `CreateWithItem()` method is provided in Chapter "Creating a device".

The following table lists the assignment of the devices and their type identifiers:

SINUMERIK NCU	Type identifier
SINUMERIK 840D sl NCU 710.3 PN	Article No.:6FC5 371-0AA30-0Axx/Vy.z
SINUMERIK 840D sl NCU 720.3 PN	Article No.:6FC5 372-0AA30-0Axx/Vy.z
SINUMERIK 840D sl NCU 730.3 PN	Article No.:6FC5 373-0AA30-0Axx/Vy.z
SINUMERIK 840D sl NCU 730.3 PN /319	Article No.:6FC5 373-0AA31-0Axx/Vy.z

When creating SINUMERIK devices, place holders in the type identifier are permissible. You can subsequently replace these placeholders by device-specific characters. If, for example, you enter type identifier "OrderNumber:6FC5 372-0AA30-0AA0/V4.8", then SINUMERIK 840D sl NCU 720.3 PN is created with firmware version FW4.8.

Classification of the device element

Every device or device element has obligatory attributes, which are read and written to. Additional information about the attributes that are supported in Openness is provided in Chapter "Functions on device elements".

The classification attributes of the device element are write protected, and not visible in the user interface of the TIA Portal.

The classification attributes have the value "DeviceItemClassification":

Value of the classification attribute	Description
DeviceItemClassifications.None (0)	No classification.
DeviceItemClassifications.CPU (1)	The device element is a CPU.
DeviceItemClassifications.HM (2)	The device element is a header module.

If you interrogate the value of the device element via the integrated PLC, then for a SINUMERIK device, this is the value "CPU (1)".

In the Openness object model, the following components act as header modules:

- SINAMICS Integrated
- NX module
- ADI4

In all other cases, the value of the classification attribute "DeviceItemClassification" is "None" (0).

Finding device elements via the "Header module" property

The following examples show how you can find the device element with the "Header module" property, for example, before you configure a telegram.

You can only read this property, but cannot set it at device elements.

Finding device elements via the "Header module" property

```
foreach (Device device in project.Devices)
{
    foreach (DeviceItem deviceItem in device.DeviceItems)
    {
        if (deviceItem.Classification == DeviceItemClassifications.HM)
        {
            var driveObjectContainer = deviceItem.GetService<DriveObjectContainer>();
            // do something
        }
    }
}
```

Alternatively, use the following example code to find a PLC, irrespective of its specific implementation (integrated SINUMERIK PLC, SIMATIC PLC, software PLC in the PC) based on the "CPU" property:

Finding PLCs

```
foreach (Device device in project.Devices)
{
    DeviceItem rack = device.DeviceItem[0]; //additional necessary step in SINUMERIK side
    foreach (DeviceItem deviceItem in rack.DeviceItems)
    {
        if (deviceItem.Classification == DeviceItemClassifications.CPU)
        {
            var softwareContainer = deviceItem.GetService<SoftwareContainer>();
            // do something
        }
    }
}
```

See also

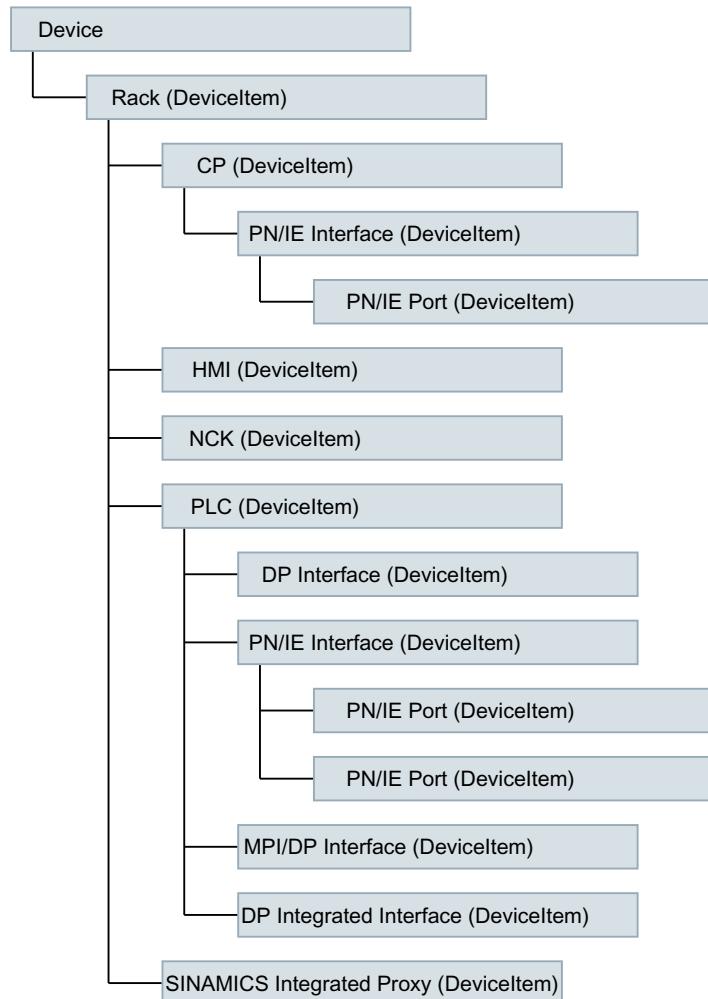
[Creating an NX module \(Page 643\)](#)

[Creating an NCU \(Page 643\)](#)

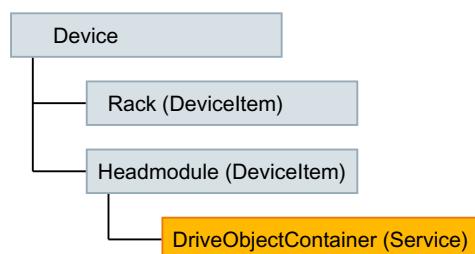
7.26.4 Object model

Overview

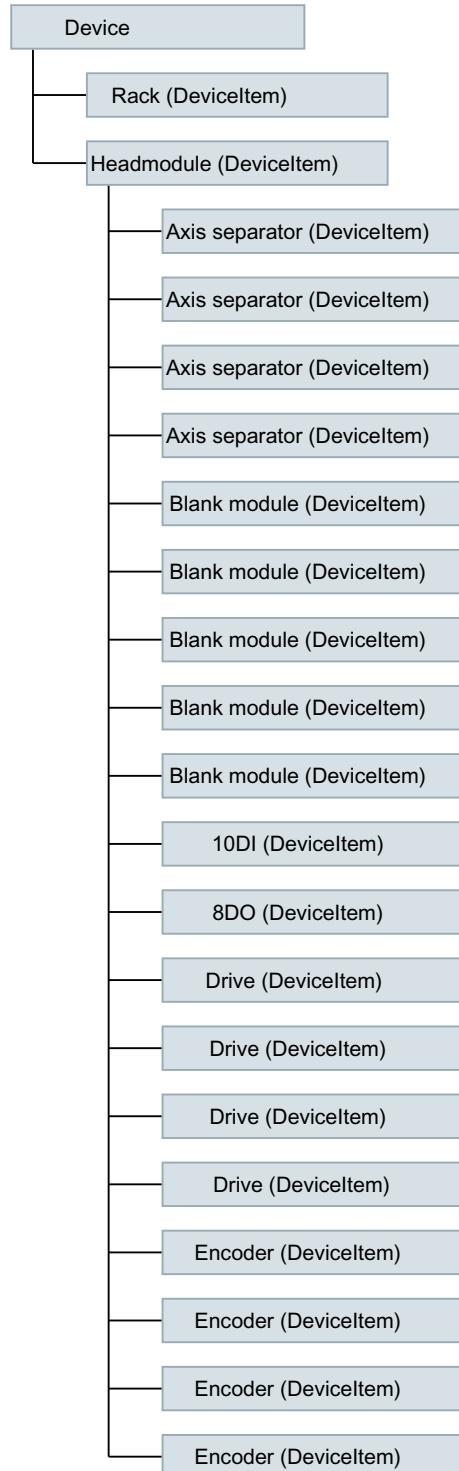
The following diagram describes the objects located under "Device" for SINUMERIK 840D sl:



The following diagram describes the objects located under "Device" for SINUMERIK NX 10.3/15.3:



The following diagram describes the objects located under "Device" for ADI4:



For more information on TIA Portal Openness object model, see Chapter "TIA Portal Openness API".

7.26.5 Reference

7.26.5.1 DriveObject

DriveObject

The `DriveObject` class allows access to the drive object. Using the drive object, the telegram can be accessed, for example.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **properties** of the class:

Name	Data type	Access mode	Access	Description
Telegrams	TelegramComposition	read	-	Returns a list with the available telegrams of the drive object. The list can be changed with the <code>TelegramComposition</code> class.
Parent	IEngineeringObject	read	-	Returns the reference to the higher-level class (<code>DriveObjectContainer</code>).

The following table describes the **methods** of the class:

Name	Description
GetAttribute	Performs read access to an attribute of a drive object
SetAttribute	Performs write access to an attribute of a drive object
GetEnumerator	Facilitates the iteration over the specified set of elements

7.26.5.2 DriveObjectContainer

DriveObjectContainer

The `DriveObjectContainer` is a service of the drive object (`DeviceItem`) for the current device (`Device`).

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **navigators** of the DriveObjectContainer:

Name	Data type	Access mode	Access	Description
DriveObjects	DriveObjectComposition	read	-	Returns a list with the available drive objects. The drive objects allow access to the drive parameters and telegrams.
Parent	IEngineeringObject	read	-	Returns the reference to the higher-level class (DeviceItem).

7.26.5.3 DriveObjectComposition

DriveObjectComposition

The DriveObjectComposition class facilitates the access to available telegrams of a drive object.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **properties** of the class:

Name	Data type	Access mode	Access	Description
IndexOf	int32	read	-	Returns the index in the set of the elements for the queried instance.
Contains	bool	read	-	Determines as to whether the specified instance is included in the set of elements. TRUE: The container contains the instance FALSE: The container does not contain the instance
GetEnumerator	IEnumerator<DriveObject>	read	-	Facilitates the iteration over the specified set of elements

7.26.5.4 AddressComposition

AddressComposition

The AddressComposition class represents the address of a telegram.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **properties** of the class:

Name	Data type	Access mode	Access	Description
IoType	enum: AddressIoType	read	-	Returns information on the type of the address.
Context	enum: AddressContext	read	Access only possible via GetAttribute or SetAttribute	Returns information on the context of the address.
StartAddress	Int32	read/write	-	Returns the start address of the telegram or defines the start address.
Length	Int32	read	-	Returns the length of the telegram.
IndexOf	int32	read	-	Returns the index in the set of the elements for the queried instance.
Contains	bool	read	-	Determines as to whether the specified instance is included in the set of elements. TRUE: The container contains the instance FALSE: The container does not contain the instance
GetEnumerator	IEnumerator<DriveObject>	read	-	Facilitates the iteration over the specified set of elements

Note

If you want to navigate back to the telegram via `Address.Parent`, instead of namespace `Siemens.Engineering.MC.Drives.Telegram`, namespace `Siemens.Engineering.MC.DriveConfiguration.Telegram` is called.

You can find further information about TIA Portal Openness libraries under "Standard libraries".

See also

[AddressIoType \(Page 642\)](#)

[AddressContext \(Page 642\)](#)

7.26.5.5 AddressContext

AddressContext

The Enum AddressContext contains information on the context of the address.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **enum entries**:

Name	Description
AddressContext.None	No context has been found for the address
AddressContext.Device	Context is a device address
AddressContext.Head	Context is a head address

You can find further information about TIA Portal Openness libraries under "Standard libraries".

7.26.5.6 AddressIoType

AddressIoType

The Enum AddressIoType contains information on the type of the address.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **enum entries**:

Name	Description
AddressIoType.None	The IO type cannot be used
AddressIoType.Input	Type is an input address
AddressIoType.Output	Type is an output address
AddressIoType.Diagnosis	Type is a diagnosis address
AddressIoType.Substitute	Type is a substitute address

You can find further information about TIA Portal Openness libraries under "Standard libraries".

7.26.6 Code example

7.26.6.1 General

The following code examples describe the basic procedure for various applications. The code is not necessarily complete or compilable.

7.26.6.2 Executing the first steps in SINUMERIK

- Connect the TIA Portal Openness application with the TIA Portal.
- Open your project.

The following example shows how you can determine which SINUMERIK Toolbox version is installed.

Determining the SINUMERIK Toolbox version

```
using Siemens.Engineering;
if (tiaProcess.InstalledSoftware.Any(sw => sw.Name.Equals("SINUMERIK Toolbox") &&
sw.Version.Equals("V16")))
{
    Console.WriteLine("SINUMERIK Toolbox is available");
}
// "V16" is the current SINUMERIK version started at December 2019.
```

7.26.6.3 Creating an NCU

You create a SINUMERIK NCU using the `CreateWithItem()` method of the `Devices` Collection. SINUMERIK NCUs are specified using method parameters. The format of the parameters is described in the following.

Creating an NCU

Parameter format for SINUMERIK NCU:

```
CreateWithItem(@"OrderNumber:mlfb/
FirmwareVersion/", "NameOfTheDevice", positionNumber)
```

The `positionNumber` parameter is optional.

The following example shows how you create a SINUMERIK 840D si NCU 720.3 PN control system.

Creating a SINUMERIK 840D si NCU 720.3 PN

```
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
Project tiaproject= portal.Projects.Open(...); //The path of the project

Device NCUDevice = tiaproject.Devices.CreateWithItem(@"OrderNumber:6FC5
372-0AA30-0AA0/4.8/", "NCU 720.3 PN", string.Empty, "TestDevice");
```

7.26.6.4 Creating an NX module

You create an NX module using the `Device CreateWithItem` method. You then connect the NX module to an NCU.

The following type identifiers are used for SINUMERIK NX modules:

SINUMERIK NX module	Type identifier
SINUMERIK NX10.3	OrderNumber:6SL3 040-1NC00-0Axx/Vy.z
SINUMERIK NX15.3	OrderNumber:6SL3 040-1NB00-0Axx/Vy.z

The following example shows how you can create a SINUMERIK NX module.

Creating an NX module

```
project.Devices.CreateWithItem("OrderNumber:6SL3040-1NC00-0AA0/
V5.1", "MyNXDevice", "TestDevice");
```

Version compatibility

The firmware version of the NX module must match the firmware version of SINAMICS Integrated - and must be compatible with the firmware version of the NCU.

An NX firmware version that deviates from SINAMICS Integrated cannot be assigned via Openness.

The following table lists the version compatibility for 840D sl:

NCU firmware (840D sl)	SINAMICS Integrated/NX firmware
V4.5	V4.5
V4.7	V4.7
V4.8	V5.1
V4.91	V5.2
V4.92	
V4.93	

7.26.6.5 Connecting an NX module with NCU

Connecting an NX module with an NCU

To connect an NX module with an NCU via subnet type "ProfibusIntegrated", the "NetworkInterface" service must be loaded via the header module of NX.

The following example shows how you can load the "NetworkInterface" service.

Loading the NetworkInterface service

```
foreach (Device device in project.Devices)
{
    foreach (DeviceItem deviceItem in device.DeviceItems)
    {
        if (deviceItem.Classification == DeviceItemClassifications.HM)
        {
            var networkInterface = deviceItem.GetService<NetworkInterface>();
            // do something
        }
    }
}
```

The following example shows how you can connect an NX module with an NCU via "ProfibusIntegrated":

Connecting an NX module via ProfibusIntegrated

```
Subnet pbiSubnet = ...;
Node node = networkInterface.Nodes.FirstOrDefault();
node.ConnectToSubnet(pbiSubnet);
```

The DP address is assigned to the fixed NX in Openness via DRIVE-CLiQ port. Each port label has a fixed DP-integrated address.

The DP address must be assigned to the NX before connection to the NCU.

7.26.6.6 Creating archives

Creating archives for series commissioning

Using TIA Portal Openness, create and export SINUMERIK archives to simplify series commissioning, for example.

Note

The PLC must be in offline mode while the archives are being created. The safety mode must not be active.

You create SINUMERIK archives via the TIA Portal project property `HwUtilities` with the `SinumerikArchiveProvider` **service**.

The following example shows how to call the `SinumerikArchiveProvider` **service**:

Calling SinumerikArchiveProvider

```
Project project = ...;
SinumerikArchiveProvider archiveProvider =
project.HwUtilities.Find("SinumerikArchiveProvider") as
SinumerikArchiveProvider;

if (archiveProvider != null)
{
// Work with the provider
}
```

The following example shows how to create a PLC archive which contains the hardware information and all of the data blocks:

Creating a PLC archive

```
...
Siemens.Engineering.HW.DeviceItem plc = ...;

try {
    // The file extension is required
    string archivePath = string.Format(@"D:\some_path\{0}.dsf", plc.Name);

    // Comment and author arguments are optional
    archiveProvider.Archive(plc, new FileInfo(archivePath),
    SinumerikArchivationMode.HardwareAndAllProgramBlocks[, "Comment", "Author
    name"]);
}
catch (EngineeringTargetInvocationException ex)
{
    // Handle archive failure
}
```

The following example shows how to update the software portions of a previously created archive:

Updating portions of archives

```
...
Siemens.Engineering.HW.DeviceItem plc_1 = ...;
Siemens.Engineering.HW.DeviceItem plc_1_copy = ...;
try {
    // The file extension is required
    string archivePath = @"D:\some_path\SinumerikArchive.dsf";

    // Create a Sinumerik archive with HardwareAndAllProgramBlocks
    archiveProvider.Export(plc_1, new FileInfo(archivePath),
    SinumerikArchivationMode.HardwareAndAllProgramBlocks);

    // Update the software part in the previously created archive using
    UpdateProgramBlocksOfArchive method
    archiveProvider.UpdateProgramBlocksOfArchive(plc_1_copy, new
    FileInfo(archivePath));
}
catch (EngineeringException ex)
{
    // Handle export failure
}
```

7.26.6.7 Activating Safety Integrated

Activating Safety Integrated

With TIA Portal Openness, you can activate Safety Integrated (F-PLC) in the properties of the NCU.

Note

Effects on telegram configuration

The Safety Integrated mode used affects the telegram configuration because telegrams other than the ones used for inactive Safety Integrated mode are used in Safety Integrated plus (F-PLC) mode.

However, added or changed telegrams are retained if they are compatible with the newly selected Safety Integrated mode.

If applicable, after the mode change in the telegram configuration, ensure that any adjustments are still available.

You activate and deactivate Safety Integrated (F-PLC) with the `SafetyModeProvider` service.

Note

The PLC must be in offline mode if Safety Integrated (F-PLC) is being activated or deactivated.

The following example shows how to call the `SafetyModeProvider` service:

Calling `SafetyModeProvider`

```
...
Siemens.Engineering.HW.Device ncu = ...;
try
{
    SafetyModeProvider provider = ncu.GetService<SafetyModeProvider>();
    //Perform the safety mode change:
    provider.SetSafetyMode(SafetyMode.DbSI);
}
catch( (EngineeringException ex) )
{
    // Handle safety mode change failure
}
```

The following example shows how to call the current Safety Integrated setting on the device:

Calling the safety setting of the device

```
...
Siemens.Engineering.HW.Device ncu = ...;
try
{
    SafetyModeProvider provider = ncu.GetService<SafetyModeProvider>();
    //Query the safety mode:
    SafetyMode safetyMode = provider.CurrentMode;
}
catch( (EngineeringException ex) )
{
    // Handle any failure
}
```

7.27 Functions for SINUMERIK ONE

7.27.1 Introduction

Using TIA Portal-Openness, you automate the engineering and control the TIA Portal using a program that you created yourself.

In this help document, you can find a lot of information and code examples for this program that you generate yourself. You can also generate and use your own programs for the TIA Portal "SINUMERIK" application.

Further information

Before you generate your own program for SINUMERIK from the sample codes listed in the following, please note the general information on Openness, which you can find in this help under the following keywords:

- Preconditions for using TIA Portal Openness
- Installing TIA Portal Openness
- Access the TIA Portal
- TIA Portal Openness object model
- Programming steps

7.27.2 Type identifier designation of the components

Each SINUMERIK component has a unique number, which is designated as type identifier (TypelIdentifier). In the Openness program code, you can use the type identifier to clearly identify and designate a component. For example, the type identifier for SINUMERIK ONE NCU 1750 "Article No.: 6FC5 317-5AA00-0Axx/Vy.z".

The type identifier is displayed when creating a device using the dialog "Add new device" and in the tabular device overview.

You can copy the type identifier into your Openness application.

The type identifier is expected in Openness as a current parameter when a method is called, e.g. the `CreateWithItem()` method.

Activating display of the type identifier in SINUMERIK

1. In the project view, choose "Options > Settings" from the menu.
The "Settings" configuration area opens.
2. In the secondary navigation, select entry "Hardware configuration".
3. Activate option "Activate display of the type identifier for devices and modules".
The type identifier display is now active.

7.27.3 Fundamentals

You can create the following SINUMERIK devices in the TIA Portal with Openness:

- NCU
- NX module
- PPU

To create the SINUMERIK devices, use the method `CreateWithItem()` of the `Devices` Collection.

Note

All integrated subcomponents of a SINUMERIK-NCU - such as PLC, NCK, CP, HMI and SINAMICS Integrated - are automatically created at the same subordinate level.

Special issue when creating SINUMERIK devices

The names of the subracks correspond to the NCU types and are write protected in the user interface. You must also use the names of the subracks in the TIA Portal Openness. Use the following standard names of the subracks for creating a device using a device element type identifier with the `Device CreateWithItem` method:

SINUMERIK device	Standard subrack names
SINUMERIK ONE NCU 1750	NCU 1750
SINUMERIK ONE NCU 1760	NCU 1760
SINUMERIK ONE PPU 1740	PPU 1740

Note

Alternatively, you can omit the parameter names. The standard name is used if the name is "null" or "String.Empty".

The following parameters can be used with the `CreateWithItem()` method:

- default name, e.g. `project.Devices.CreateWithItem("OrderNumber:6FC5 317-5AA00-0AA0/V6.13", "NCU 1750", "TestDevice");`
- null, e.g. `project.Devices.CreateWithItem("OrderNumber:6FC5 317-5AA00-0AA0/V6.13", null, "TestDevice");`
- `string.Empty`, e.g. `project.Devices.CreateWithItem("OrderNumber:6FC5 317-5AA00-0AA0/V6.13", string.Empty, "TestDevice");`

Additional information about the call parameters associated with the `CreateWithItem()` method is provided in Chapter "Creating a device".

The following table lists the assignment of the devices and their type identifiers:

SINUMERIK device	Type identifier
SINUMERIK ONE NCU 1750	Article No.: 6FC5 317-5AA00-0Axx/Vy.z
SINUMERIK ONE NCU 1760	Article No.: 6FC5 317-6AA00-0Axx/Vy.z
SINUMERIK ONE PPU 1740	Article No.: 6FC5 317-4AA00-1xxx/Vy.z

When creating SINUMERIK devices, place holders in the type identifier are permissible. You can subsequently replace these placeholders by device-specific characters.

Classification of the device element

Every device or device element has obligatory attributes, which are read and written to. Additional information about the attributes that are supported in Openness is provided in Chapter "Functions on device elements".

The classification attributes of the device element are write protected, and not visible in the user interface of the TIA Portal.

The classification attributes have the value "DeviceItemClassification":

Value of the classification attribute	Description
<code>DeviceItemClassifications.None (0)</code>	No classification.
<code>DeviceItemClassifications.CPU (1)</code>	The device element is a CPU.
<code>DeviceItemClassifications.HM (2)</code>	The device element is a header module.

If you interrogate the value of the device element via the integrated PLC, then for a SINUMERIK device, this is the value "CPU (1)".

In the Openness object model, the following components act as header module:

- SINAMICS Integrated
- NX module

In all other cases, the value of the classification attribute is "DeviceItemClassification" "None" (0).

Finding device elements via the "Header module" property

The following examples show how you can find the device element with the "Header module" property, for example, before you configure a telegram.

You can only read this property, but cannot set it at device elements.

Finding Sinamics Integrated / NX via the "Header module" property

```
foreach (Device device in project.Devices)
{
    foreach (DeviceItem deviceItem in device.DeviceItems)
    {
        if (deviceItem.Classification == DeviceItemClassifications.HM)
        {
            var driveObjectContainer = deviceItem.GetService<DriveObjectContainer>();
            // do something
        }
    }
}
```

Alternatively, use the following example code to find a PLC, irrespective of its specific implementation (integrated SINUMERIK PLC, SIMATIC PLC, software PLC in the PC) based on the "CPU" property:

Finding PLCs

```
Device ncuDevice = ...
DeviceItem plc = GetPlc(ncuDevice.DeviceItems);

...
DeviceItem GetPlc(DeviceItemComposition deviceItems)
{
    if (deviceItems.Count == 0)
    {
        return null;
    }
    foreach (var deviceItem in deviceItems)
    {
        if (deviceItem.Classification == DeviceItemClassifications.CPU)
            return deviceItem;

        DeviceItem plc = GetPlc(deviceItem.DeviceItems);
        if (plc != null)
            return plc;
    }

    return null;
}
```

See also

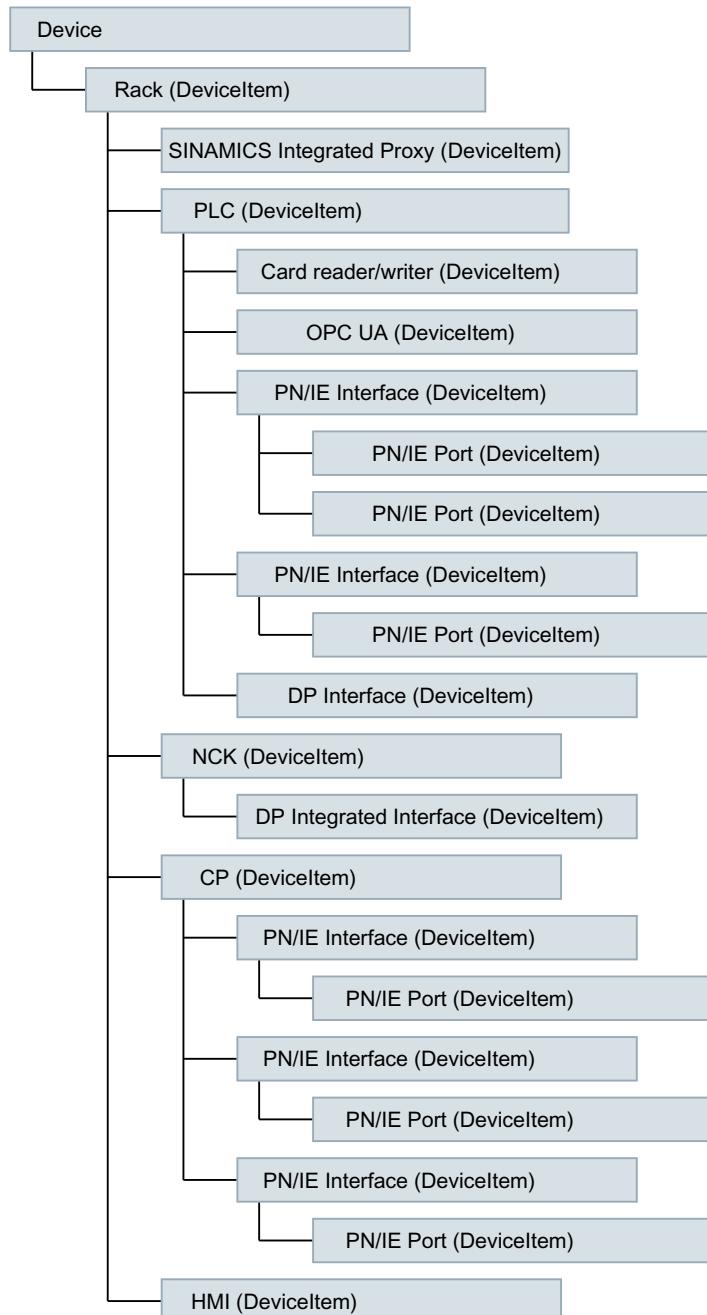
[Creating an NCU \(Page 670\)](#)

[Creating an NX module \(Page 670\)](#)

7.27.4 Object model

Overview

The following diagram describes the objects located under "Device" for SINUMERIK ONE:



For more information on the TIA Portal Openness object model, see Chapter "TIA Portal Openness API".

Available namespaces for telegram configuration

The following namespaces are valid in SINUMERIK Openness for the telegram configuration interfaces:

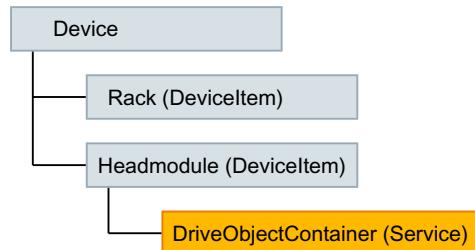
Namespace	Assembly
Siemens.Engineering.MC.DriveConfiguration (Page 654)	Siemens.Engineering.MC.DriveConfiguration in Siemens.Engineering.dll
Siemens.Engineering.MC.Drives (Page 660)	Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Note

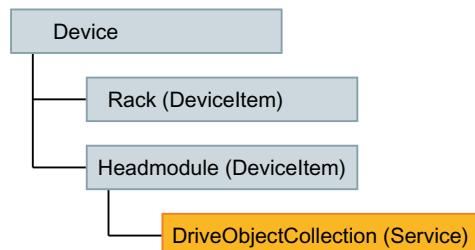
With the namespace `Siemens.Engineering.MC.DriveConfiguration`, additional functions are available, e.g. you can create, delete or reorder drive objects.

In the long term, the namespace `Siemens.Engineering.MC.DriveConfiguration` remains. However, the namespace `Siemens.Engineering.MC.Drives` will still be supported into the next versions of Openness for compatibility reasons.

The following diagram describes the objects in SINUMERIK NX 10.3/15.3 that are located under "Device" in the namespace `Siemens.Engineering.MC.Drives`:



The following diagram describes the objects in SINUMERIK NX 10.3/15.3 that are located under "Device" in the namespace `Siemens.Engineering.MC.Drives` and `Siemens.Engineering.MC.DriveConfiguration`:



7.27.5 Reference

7.27.5.1 Namespace Siemens.Engineering.MC.DriveConfiguration

DriveObject

DriveObject

The `DriveObject` class allows access to the drive object. Using the drive object, the telegram can be accessed, for example.

Namespace: Siemens.Engineering.MC.DriveConfiguration

Assembly: Siemens.Engineering.MC.DriveConfiguration
in Siemens.Engineering.dll

The following table describes the **properties** of the class:

Name	Data type	Access mode	Access	Description
Telegrams	TelegramComposition	read	-	Returns a list with the available telegrams of the drive object. The list can be changed with the <code>TelegramComposition</code> class.
Parent	IEngineeringObject	read	-	Returns the reference to the higher-level class (<code>DriveObjectCollection</code> and <code>DriveObjectContainer</code>).
Category		read	-	Returns the reference to the higher-level class (<code>DriveObjectCategory</code>)
Name		read	-	Returns the name of the drive object.

The following table describes the **methods** of the class:

Name	Description
GetAttribute	Performs read access to an attribute of a drive object
GetAttributes	Performs read access to all attributes of a drive object
GetAttributeInfos	Performs read access to information on attributes of a drive object
SetAttribute	Performs write access to an attribute of a drive object
SetAttributes	Performs write access to all attributes of a drive object
Delete	Deletes the drive object instance at which the "Delete" is called

DriveObjectCollection

DriveObjectCollection

The `DriveObjectCollection` is a service of the drive object (`DeviceItem`) of the current device (`Device`).

Namespace: Siemens.Engineering.MC.DriveConfiguration
Assembly: Siemens.Engineering.MC.DriveConfiguration
 in Siemens.Engineering.dll

The following table describes the **navigators** of the `DriveObjectCollection`:

Name	Data type	Access mode	Access	Description
DriveObjects	DriveObjectComposition	read	-	Returns a list with the available drive objects. The drive objects allow access to the drive parameters and telegrams.

The following table describes the **methods** of the class:

Name	Description
GetAttribute	Performs read access to an attribute of <code>DriveObjectCollection</code>
GetAttributes	Performs read access to all attributes of <code>DriveObjectCollection</code>
GetAttributeInfos	Performs read access to information on attributes of <code>DriveObjectCollection</code>
SetAttribute	Performs write access to an attribute of <code>DriveObjectCollection</code>
SetAttributes	Performs write access to all attributes of <code>DriveObjectCollection</code>

DriveObjectComposition

DriveObjectComposition

The `DriveObjectComposition` class allows access to available telegrams of a drive object and contains all of the drive objects of an NCU or NX module.

Namespace: Siemens.Engineering.MC.DriveConfiguration
Assembly: Siemens.Engineering.MC.DriveConfiguration
 in Siemens.Engineering.dll

The following table describes the **properties** of the class:

Name	Data type	Access mode	Access	Description
Parent	IEngineeringObject	read	-	Returns the reference to the higher-level class (DriveObjectContainer)
Count		read	-	
IsReadOnly		read	-	

The following table describes the **methods** of the class:

Name	Description
GetEnumerator	Facilitates the iteration over the specified set of elements
Contains	Determines whether the specified instance is contained in the set of the elements.TRUE: The container contains the instance FALSE: The container does not contain the instance
IndexOf	Returns the index in the set of the elements for the queried instance.
Create	Creates a DriveObjectComposition class
Find	Searches for a DriveObjectComposition class

DriveObjectCategory

DriveObjectCategory

The enum `DriveObjectCategory` contains predefined categories of the drive objects. Read access to `DriveObjectCategory` is possible.

Namespace: Siemens.Engineering.MC.DriveConfiguration

Assembly: Siemens.Engineering.MC.DriveConfiguration
in Siemens.Engineering.dll

The following table contains the predefined categories of the drive objects:

Supported drive objects in SINUMERIK	DriveObjectCategory
SinumerikDriveAxis	SERVO, ENC, HLA, TM41
SinumerikControlUnit	CU_I, CU_NX_CX
SinumerikInfeed	A_INF, B_INF, S_INF

TelegramComposition

TelegramComposition

The `TelegramComposition` class facilitates the access to the telegrams of a drive object (`DriveObject` (Page 654)). The structure of a telegram can be read out via the `Telegram` (Page 658) class.

Changes to telegram objects (e.g. changing the safety telegram) can result in the relevant telegram object being deleted in the `TelegramComposition` or a new telegram object being created. In this case, you must search through the `TelegramComposition` again to find the new telegram object (Page 658) again after the change.

If the respective drive object does not support telegrams, the value for `TelegramComposition` is returned blank.

Namespace: Siemens.Engineering.MC.DriveConfiguration

Assembly: Siemens.Engineering.MC.DriveConfiguration
in Siemens.Engineering.dll

The following table describes the **properties** of the class:

Name	Data type	Access mode	Access	Description
Parent	IEngineeringObject	read	-	Returns the reference to the higher-level class (<code>DriveObject</code>).
Count		read	-	
IsReadOnly		read	-	

The following table describes the **methods** of the class:

Name	Description
Create (enum TelegramId Id)	Creates a telegram; the type of the telegram corresponds to the ID.
int IndexOf (TelegramType)	Returns the index in the set of the elements for the queried instance.
bool Contains	Determines as to whether the specified instance is included in the set of elements. TRUE: The container contains the instance FALSE: The container does not contain the instance
Contains	Determines whether the specified instance is contained in the set of the elements. TRUE: The container contains the instance FALSE: The container does not contain the instance
IEnumerator GetEnumerator	Allows <code>IEnumerator<DriveObject></code> the iteration via the given set of the elements

Telegram

Telegram

The **Telegram** class allows access to the structure of a telegram from a drive object.

Namespace: Siemens.Engineering.MC.DriveConfiguration

Assembly: Siemens.Engineering.MC.DriveConfiguration
in Siemens.Engineering.dll

The following table describes the **properties** of the class:

Name	Data type	Access mode	Access	Description
Id	Int32	read	-	Returns the ID of the telegram.
Type	enum:TelegramType (Page 659)	read	-	Returns the type of telegram as Enum TelegramType.
Addresses	AddressComposition (Page 667)	read	-	Returns an AdressComposition with information on the address.

The following table describes the **methods** of the class:

Name	Description
GetSize(AddressIoType (Page 668))	Returns the size of the inputs or outputs of the telegram.
CanChangeSize(AddressIoType (Page 668), Int32, bool)	Returns <code>true</code> if the size of the telegram can be changed as parameterized. Standard telegrams can only be enlarged. The retention of the previous telegram address is taken into account when the option is parameterized with <code>true</code> .
CanSetSize (AddressIoType (Page 668) Int32, bool)	Returns true if the size of the telegram can be changed as parameterized.
SetSize	Performs write access to the telegram size
Delete	Deletes the telegram instance at which "Delete" is called
GetAttribute	Performs read access to a telegram attribute
GetAttributes	Performs read access to all telegram attributes
GetAttributeInfos	Performs read access to information on telegram attributes
SetAttribute	Performs write access to a telegram attribute
SetAttributes	Performs write access to all telegram attributes

Properties of Safety telegrams

The following table describes the additional **properties** of the "Telegram" class of the type "SafetyTelegram".

Name	Data type	Access mode	Access	Description
Failsafe_FSourceAddress	UInt32	read	Dynamic	PROFIsafe source address
Failsafe_FDestinationAddress	UInt32	read/write	Dynamic	PROFIsafe destination address
Failsafe_FIODBNumber	UInt32	read/write	Dynamic	Safety DB number, write access only possible if the property "Failsafe_ManualAssignmentFIODBNu mber" is assigned the value "1"
Failsafe_FIODBName	string	read	Dynamic	Name Safety DB
Failsafe_ManualAssignmentFIODBNu mber	bool	read/write	Dynamic	Setting for manual assignment of the prop erty "Failsafe_FIODBNumber"
Failsafe_FMonitoringtime	UInt32	read/write	Dynamic	Safety monitoring time, write access only possible if the property "Failsafe_ManualAssignmentFMoni toringtime" is set to "true"
Failsafe_ManualAssignmentFMonito ringtime	bool	read/write	Dynamic	Setting for manual assignment of the prop erty "Failsafe_FMonitoringtime"

The additional properties are accessed via `GetAttribute` and `SetAttribute`, e.g. `GetAttribute("Failsafe_FSourceAddress")`. `UInt32` is expected as return value.

TelegramType

TelegramType

The Enum `TelegramType` contains predefined telegram types.

Namespace: Siemens.Engineering.MC.DriveConfiguration
Assembly: Siemens.Engineering.MC.DriveConfiguration
 in Siemens.Engineering.dll

The following table describes the enum entries:

Name	Description
MainTelegram	Main telegram: e.g. telegram 136
SupplementaryTelegram	Supplementary telegram: e.g. telegram 701
AdditionalTelegram	Additional telegram: free telegram
SafetyTelegram	Safety telegram: e.g. telegram 903

Note

The torque telegram (`TorqueTelegram`) is not supported, although it is available in the TIA Portal Openness API in the SINUMERIK context.

TelegramId

TelegramId

The enum `TelegramId` contains the telegram numbers that are relevant for communication between the PLC and the drive. The IDs are defined via the PROFdrive standard.

Namespace: Siemens.Engineering.MC.DriveConfiguration
Assembly: Siemens.Engineering.MC.DriveConfiguration
 in Siemens.Engineering.dll

7.27.5.2 Namespace Siemens.Engineering.MC.Drives

DriveObject

DriveObject

The `DriveObject` class allows access to the drive object. Using the drive object, the telegram can be accessed, for example.

Namespace: Siemens.Engineering.MC.Drives
Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **properties** of the class:

Name	Data type	Access mode	Access	Description
Telegrams	TelegramComposition	read	-	Returns a list with the available telegrams of the drive object. The list can be changed with the <code>TelegramComposition</code> class.
Parent	IEngineeringObject	read	-	Returns the reference to the higher-level class (<code>DriveObjectContainer</code>).

The following table describes the **methods** of the class:

Name	Description
GetAttribute	Performs read access to an attribute of a drive object
SetAttribute	Performs write access to an attribute of a drive object
GetEnumerator	Facilitates the iteration over the specified set of elements

See also

[TelegramComposition \(Page 662\)](#)

DriveObjectContainer

DriveObjectContainer

The `DriveObjectContainer` is a service of the drive object (`DeviceItem`) for the current device (`Device`).

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **navigators** of the `DriveObjectContainer`:

Name	Data type	Access mode	Access	Description
DriveObjects	DriveObjectComposition	read	-	Returns a list with the available drive objects. The drive objects allow access to the drive parameters and telegrams.
Parent	IEngineeringObject	read	-	Returns the reference to the higher-level class (<code>DeviceItem</code>).

DriveObjectCollection

DriveObjectCollection

The `DriveObjectCollection` is a service of the drive object (`DeviceItem`) of the current device (`Device`).

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **navigators** of the `DriveObjectCollection`:

Name	Data type	Access mode	Access	Description
DriveObjects	DriveObjectComposition	read	-	Returns a list with the available drive objects. The drive objects allow access to the drive parameters and telegrams.
Parent	IEngineeringObject	read	-	Returns the reference to the higher-level class (<code>DeviceItem</code>).

DriveObjectComposition

DriveObjectComposition

The `DriveObjectComposition` class facilitates the access to available telegrams of a drive object.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **properties** of the class:

Name	Data type	Access mode	Access	Description
IndexOf	int32	read	-	Returns the index in the set of the elements for the queried instance.
Contains	bool	read	-	Determines as to whether the specified instance is included in the set of elements. TRUE: The container contains the instance FALSE: The container does not contain the instance
GetEnumerator	IEnumerator<DriveObject>	read	-	Facilitates the iteration over the specified set of elements

TelegramComposition

TelegramComposition

The `TelegramComposition` class facilitates the access to the telegrams of a drive object (`DriveObject` (Page 660)). The structure of a telegram can be read out via the `Telegram` (Page 664) class.

Changes to telegram objects (e.g. changing the safety telegram) can result in the relevant telegram object being deleted in the `TelegramComposition` or a new telegram object being created. In this case, you must search through the `TelegramComposition` again to find the new telegram object (Page 664) again after the change.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **methods** of the class:

Name	Description
CanInsertAdditionalTelegram(Int32, Int32)	Returns <code>true</code> if an extension can be created in accordance with the parameterized sizes (input and output sizes).
InsertAdditionalTelegram(Int32, Int32)	Creates an extension for the drive object in accordance with the parameterized sizes and returns <code>true</code> if the extension could be inserted. In the event of an error, an <code>EngineeringTargetInvocationException</code> is triggered.
CanInsertSupplementaryTelegram(Int32)	Returns <code>true</code> if a supplementary telegram can be created in accordance with the parameterized telegram number.
InsertSupplementaryTelegram(Int32)	Creates the supplementary telegram with the parameterized telegram number and returns <code>true</code> if the telegram could be inserted. In the event of an error, an <code>EngineeringTargetInvocationException</code> is triggered.
CanInsertSafetyTelegram(Int32)	Returns <code>true</code> if a safety telegram can be generated corresponding to the parameterized telegram number.
InsertSafetyTelegram(Int32)	Creates the supplementary telegram with the parameterized telegram number and returns <code>true</code> if the telegram could be inserted. In the event of an error, an <code>EngineeringTargetInvocationException</code> is triggered.
EraseTelegram(TelegramType)	Returns <code>true</code> if the parameterized telegram could be deleted. Standard telegrams cannot be deleted. In the event of an error, an <code>EngineeringTargetInvocationException</code> is triggered.
Find(TelegramType)	Returns the <code>Telegram</code> (Page 664) object if it could be found via the parameterized telegram type. <code>null</code> if the telegram is not found. Example <code>Telegram telegram = telegrams.Find(TelegramType.MainTelegram);</code>
int IndexOf(TelegramType)	Returns the index in the set of the elements for the queried instance.

Name	Description
Parent	Returns the reference to the higher-level class (<code>DriveObject</code>).
bool Contains	Determines as to whether the specified instance is included in the set of elements. TRUE: The container contains the instance FALSE: The container does not contain the instance
IEnumerator GetEnumerator	Facilitates <code>IEnumerator<DriveObject></code> the iteration over the specified set of elements

Telegram

Telegram

The `Telegram` class allows access to the structure of a telegram from a drive object.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **properties** of the class:

Name	Data type	Access mode	Access	Description
TelegramNumber	Int32	read	-	Returns the number of the main telegram or specifies the number. A freely configurable telegram has the number 999.
Type	enum: <code>TelegramType</code> (Page 666)	read	-	Returns the type of telegram as <code>Enum TelegramType</code> .
Addresses	AddressComposition (Page 667)	read	-	Returns an <code>AddressComposition</code> with information on the address.

The following table describes the **methods** of the class:

Name	Description
<code>CanChangeTelegram(Int32)</code>	Returns <code>true</code> if the telegram can be changed to the parameterized standard type.
<code>GetSize(AddressIoType (Page 668))</code>	Returns the size of the inputs or outputs of the telegram.
<code>CanChangeSize(AddressIoType (Page 668), Int32, bool)</code>	Returns <code>true</code> if the size of the telegram can be changed as parameterized. Standard telegrams can only be enlarged. The retention of the previous telegram address is taken into account when the option is parameterized with <code>true</code> .

Name	Description
ChangeSize(AddressIoType (Page 668), Int32, bool)	Returns true if the size of the telegram could be changed as parameterized. The retention of the previous telegram address is taken into account when the option is parameterized with true.
GetEnumerator	Facilitates the iteration over the specified set of elements

Properties of Safety telegrams

The following table describes the additional **properties** of the "Telegram" class of the type "SafetyTelegram".

Name	Data type	Access mode	Access	Description
Failsafe_FSourceAddress	UInt32	read	Access only possible via GetAttribute or SetAttribute	PROFIsafe source address
Failsafe_FDestinationAddress	UInt32	read/write	Access only possible via GetAttribute or SetAttribute	PROFIsafe destination address
Failsafe_FIODBNumber	UInt32	read/write	Access only possible via GetAttribute or SetAttribute	Safety DB number, write access only possible if the property "Failsafe_ManualAssignmentFIODBNumber" is assigned the value "1"
Failsafe_FIODBName	string	read	Access only possible via GetAttribute or SetAttribute	Name Safety DB
Failsafe_ManualAssignmentFIODBNumber	bool	read/write	Access only possible via GetAttribute or SetAttribute	Setting for manual assignment of the property "Failsafe_FIODBNumber"

Name	Data type	Access mode	Access	Description
Failsafe_FMonitoringtime	UInt32	read/write	Access only possible via GetAttribute or SetAttribute	Safety monitoring time, write access only possible if the property "Failsafe_ManualAssignmentFMonitoringtime" is set to "true"
Failsafe_ManualAssignmentFMonitoringtime	bool	read/write	Access only possible via GetAttribute or SetAttribute	Setting for manual assignment of the property "Failsafe_FMonitoringtime"

The additional properties are accessed via GetAttribute and SetAttribute, e.g. GetAttribute("Failsafe_FSourceAddress"). UInt32 is expected as return value.

TelegramType

TelegramType

The Enum TelegramType contains predefined telegram types.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the enum entries:

Name	Description
MainTelegram	Main telegram: e.g. telegram 136
SupplementaryTelegram	Supplementary telegram: e.g. telegram 701
AdditionalTelegram	Additional telegram: free telegram
SafetyTelegram	Safety telegram: e.g. telegram 903

Note

The torque telegram (TorqueTelegram) is not supported, although it is available in the TIA Portal Openness API in the SINUMERIK context.

AddressComposition

AddressComposition

The AddressComposition class represents the address of a telegram.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the properties of the class:

Name	Data type	Access mode	Access	Description
IoType	enum: AddressIoType	read	-	Returns information on the type of the address.
Context	enum: AddressContext	read	Access only possible via GetAttribute or SetAttribute	Returns information on the context of the address.
StartAddress	Int32	read/write	-	Returns the start address of the telegram or defines the start address.
Length	Int32	read	-	Returns the length of the telegram.
IndexOf	int32	read	-	Returns the index in the set of the elements for the queried instance.
Contains	bool	read	-	Determines as to whether the specified instance is included in the set of elements. TRUE: The container contains the instance FALSE: The container does not contain the instance
GetEnumerator	IEnumerator<DriveObject>	read	-	Facilitates the iteration over the specified set of elements

Note

If you want to navigate back to the telegram via Address.Parent, instead of namespace Siemens.Engineering.MC.Drives.Telegram, namespace Siemens.Engineering.MC.DriveConfiguration.Telegram is called.

You can find further information about TIA Portal Openness libraries under "Standard libraries".

See also[AddressIoType \(Page 668\)](#)[AddressContext \(Page 668\)](#)**AddressContext****AddressContext**

The Enum AddressContext contains information on the context of the address.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **enum entries**:

Name	Description
AddressContext.None	No context has been found for the address
AddressContext.Device	Context is a device address
AddressContext.Head	Context is a head address

You can find further information about TIA Portal Openness libraries under "Standard libraries".

AddressIoType**AddressIoType**

The Enum AddressIoType contains information on the type of the address.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **enum entries**:

Name	Description
AddressIoType.None	The IO type cannot be used
AddressIoType.Input	Type is an input address
AddressIoType.Output	Type is an output address
AddressIoType.Diagnosis	Type is a diagnosis address
AddressIoType.Substitute	Type is a substitute address

You can find further information about TIA Portal Openness libraries under "Standard libraries".

7.27.5.3 ArchiveProvider

ArchiveProvider

The ArchiveProvider class is used to generate the PLC archives.

Namespace: Siemens.Engineering.HW.Utilities
Assembly: Siemens.Engineering.HW.Utilities
 in Siemens.Engineering.dll

The following table describes the methods of the ArchiveProvider class:

Name	Parameter	Description
Archive	DeviceItem plc FileInfo path Siemens.Engineering.MC.Sinumerik.SinumerikA rchivationMode String comment (optional) String author (optional)	Creates a PLC archive

Note

If safety is activated, you cannot generate any archive via Openness.

7.27.6 Code example

7.27.6.1 General

The following code examples describe the basic procedure for various applications. The code is not necessarily complete or compilable.

7.27.6.2 Executing the first steps in SINUMERIK

- Connect the TIA Portal Openness application with the TIA Portal.
- Open your project.

The following example shows how you can determine which SINUMERIK Toolbox version is installed.

Determining the SINUMERIK Toolbox version

using Siemens.Engineering;

Determining the SINUMERIK Toolbox version

```
if (tiaProcess.InstalledSoftware.Any(sw => sw.Name.Equals("SINUMERIK Toolbox") &&
sw.Version.Equals("V16")))
{
    Console.WriteLine("SINUMERIK Toolbox is available");
}
// "V16" is the current SINUMERIK version started at December 2019.
```

7.27.6.3 Creating an NCU

You create a SINUMERIK NCU using the `CreateWithItem()` method of the `Devices Collection`. SINUMERIK NCUs are specified using method parameters. The format of the parameters is described in the following.

Creating an NCU

Parameter format for SINUMERIK NCU:

```
CreateWithItem(@"OrderNumber:mlfb/FirmwareVersion/",
"StandardSubrackName", "NameOfTheDevice")
```

The following example shows how you can create a SINUMERIK ONE NCU 1750 module.

Creating a SINUMERIK ONE NCU 1750 module

```
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
Project tiaproject= portal.Projects.Open(...); //The path of the project

Device NCUDevice = tiaproject.Devices.CreateWithItem(@"OrderNumber:6FC5
317-5AA00-0AA0/6.13/", "NCU 1750", "TestDevice");
```

7.27.6.4 Creating an NX module

You create an NX module using the `Device CreateWithItem` method. You then connect the NX module to an NCU.

The following type identifiers are used for SINUMERIK NX modules:

SINUMERIK NX module	Type identifier
SINUMERIK NX10.3	OrderNumber:6SL3 040-1NC00-0Axx/Vy.z
SINUMERIK NX15.3	OrderNumber:6SL3 040-1NB00-0Axx/Vy.z

The following example shows how you can create a SINUMERIK NX module.

Creating an NX module

```
project.Devices.CreateWithItem("OrderNumber:6SL3040-1NC00-0AA0/
V5.2", "MyNXDevice", "TestDevice");
```

Version compatibility

The firmware version of the NX module must match the firmware version of SINAMICS Integrated - and must be compatible with the firmware version of the NCU.

An NX firmware version that deviates from SINAMICS Integrated cannot be assigned via Openness.

The following table lists the version compatibility for SINUMERIK ONE:

NCU firmware (SINUMERIK ONE)	SINAMICS Integrated/NX firmware
V6.13	V5.2

7.27.6.5 Connecting an NX module with NCU

Connecting an NX module with an NCU

To connect an NX module with an NCU via subnet type "ProfibusIntegrated", the "NetworkInterface" service must be loaded via the header module of NX.

The following example shows how you can load the "NetworkInterface" service.

Loading the NetworkInterface service

```
foreach (Device device in project.Devices)
{
    foreach (DeviceItem deviceItem in device.DeviceItems)
    {
        if (deviceItem.Classification == DeviceItemClassifications.HM)
        {
            var networkInterface = deviceItem.GetService<NetworkInterface>();
            // do something
        }
    }
}
```

The following example shows how you can connect an NX module with an NCU via "ProfibusIntegrated":

Connecting an NX module via ProfibusIntegrated

```
Subnet pbiSubnet = ...;
Node node = networkInterface.Nodes.FirstOrDefault();
node.ConnectToSubnet(pbiSubnet);
```

The DP address is assigned to the fixed NX in Openness via DRIVE-CLiQ port. Each port label has a fixed DP-integrated address.

The DP address must be assigned to the NX before connection to the NCU.

7.27.6.6 Accessing NCK events

NCK events

According to the SINUMERIK object model (Page 652), the NCK module is a `DeviceItem`.

With the following attributes of the NCK module, you can access the NCK events and configure the NCK events in TIA Portal Openness:

Name	Data type	Access mode	Description
HardwareInterruptNckToPlcSignalExchangeActive	Boolean	r/w	Activating/deactivating the event
HardwareInterruptNckToPlcSignalExchangeEventName	String	r/w	Event name
HardwareInterruptNckToPlcSignalExchangeInterrupt	Siemens.Engineering.SW.Blocks.OB	r/w	The OB assigned to the event
HardwareInterruptNckToPlcSignalExchangePriority	Int32	r/w	Event priority

All NCK events are triggered via `HardwareInterrupt` (hardware interrupt OB). The hardware interrupt OBs interrupt the cyclical program processing due to a hardware event.

The following example shows how to determine the event name for an NCK module:

Determining the event name for an NCK module

```
DeviceItem nck = ...;
string eventName =
(string)nck.GetAttribute("HardwareInterruptNckToPlcSignalExchangeEventName");
```

The following example shows how to set the hardware interrupt:

Setting the hardware interrupt

```
... DeviceItem nck = ...;
OB ob40 = ... try
{
  nck.SetAttribute("HardwareInterruptNckToPlcSignalExchangeInterrupt", ob40);
}
catch( (EngineeringException ex) )
{
// Handle setting failure
}
```

7.27.6.7 Creating archives

Creating archives for series commissioning

Using TIA Portal Openness, create and export SINUMERIK archives to simplify series commissioning, for example.

Note

The PLC must be in offline mode while the archives are being created. The safety mode must not be active.

You create SINUMERIK archives via the TIA Portal project property `HwUtilities` with the `SinumerikArchiveProvider` service.

The following example shows how to call the `SinumerikArchiveProvider` service:

Calling SinumerikArchiveProvider

```
Project project = ...;
SinumerikArchiveProvider archiveProvider =
project.HwUtilities.Find("SinumerikArchiveProvider") as SinumerikArchiveProvider;

if (archiveProvider != null)
{
// Work with the provider
}
```

The following example shows how to create a PLC archive which contains the hardware information and all of the data blocks:

Creating a PLC archive

```
...
Siemens.Engineering.HW.DeviceItem plc = ...;

try {
    // The file extension is required
    string archivePath = string.Format(@"D:\some_path\{0}.dsf", plc.Name);

    // Comment and author arguments are optional
    archiveProvider.Archive(plc, new FileInfo(archivePath),
    SinumerikArchivationMode.HardwareAndAllProgramBlocks[, "Comment", "Author name"]);
}
catch (EngineeringTargetInvocationException ex)
{
// Handle archive failure
}
```

The following example shows how to update the software portions of a previously created archive:

Updating portions of archives

```
...
Siemens.Engineering.HW.DeviceItem plc_1 = ...;
Siemens.Engineering.HW.DeviceItem plc_1_copy = ...;
try {
    // The file extension is required
    string archivePath = @"D:\some_path\SinumerikArchive.dsf";

    // Create a Sinumerik archive with HardwareAndAllProgramBlocks
    archiveProvider.Export(plc_1, new FileInfo(archivePath),
    SinumerikArchivationMode.HardwareAndAllProgramBlocks);

    // Update the software part in the previously created archive using
    UpdateProgramBlocksOfArchive method
    archiveProvider.UpdateProgramBlocksOfArchive(plc_1_copy, new FileInfo(archivePath));
}
catch (EngineeringException ex)
{
    // Handle export failure
}
```

7.27.6.8 Activating Safety Integrated

Activating Safety Integrated

With TIA Portal Openness, you can activate Safety Integrated (F-PLC) in the properties of the NCU.

Note

Effects on telegram configuration

The Safety Integrated mode used affects the telegram configuration because telegrams other than the ones used for inactive Safety Integrated mode are used in Safety Integrated plus (F-PLC) mode.

However, added or changed telegrams are retained if they are compatible with the newly selected Safety Integrated mode.

If applicable, after the mode change in the telegram configuration, ensure that any adjustments are still available.

You activate and deactivate Safety Integrated (F-PLC) with the `SafetyModeProvider` service.

Note

The PLC must be in offline mode if Safety Integrated (F-PLC) is being activated or deactivated.

The following example shows how to call the `SafetyModeProvider` service:

Calling SafetyModeProvider

```
...
Siemens.Engineering.HW.Device ncu = ...;
try
{
    SafetyModeProvider provider = ncu.GetService<SafetyModeProvider>();
    //Perform the safety mode change:
    provider.SetSafetyMode(SafetyMode.DbSI);
}
catch( (EngineeringException ex) )
{
    // Handle safety mode change failure
}
```

The following example shows how to call the current Safety Integrated setting on the device:

Calling the safety setting of the device

```
...
Siemens.Engineering.HW.Device ncu = ...;
try
{
    SafetyModeProvider provider = ncu.GetService<SafetyModeProvider>();
    //Query the safety mode:
    SafetyMode safetyMode = provider.CurrentMode;
}
catch( (EngineeringException ex) )
{
    // Handle any failure
}
```

7.27.6.9 Examples for Namespace Siemens.Engineering.MC.DriveConfiguration

Preparing telegrams

The drive communication of a SINUMERIK NCU is realized using telegrams via the SINAMICS Integrated subcomponent and, if applicable, via additionally connected NX modules.

Note

The SINUMERIK NCU and a SINAMICS Integrated are located at the same level in the TIA Portal Openness object model, and appear under "DeviceComposition" as two different devices.

Use "DriveObjectCollection" to configure a telegram. "DriveObjectCollection" is a service of the drive object (device element) of the actual header module (device element).

To start the "DriveObjectCollection" service, navigate to SINAMICS Integrated or to the header module of the NX module. The hierarchy between devices and device elements is identical for SINAMICS Integrated and NX modules.

The following example shows how to find "DriveObjectCollection" via the "Header module" property:

Finding DriveObjectCollection via header module

```
foreach (Device device in project.Devices)
{
    foreach (DeviceItem deviceItem in device.DeviceItems)
    {
        if (deviceItem.Classification == DeviceItemClassifications.HM)
        {
            var driveObjectCollection = deviceItem.GetService<DriveObjectCollection>();
            // do something
        }
    }
}
```

The SINUMERIK NCU contains a SINAMICS Integrated proxy object with references to SINAMICS Integrated.

To access a SINAMICS Integrated device or an NX module, from the SINUMERIK NCU navigate to the DP Integrated interface via NCK. Then determine the PROFIBUS master system and navigate to the connected slave.

Inserting and removing telegrams

The following example indicates how you can insert a telegram. You require a drive object for access.

You distinguish between the telegram types using their IDs.

Inserting telegrams and accessing telegram attributes

```
using Siemens.Engineering.MC.DriveConfiguration;
```

Inserting telegrams and accessing telegram attributes

```
TelegramComposition telegrams = drvObj.Telegrams;

//Create telegram
const int tgrmId = 136;
drvObj.Telegrams.CreateTelegram(tgrmId);

//Create safety telegram
const int tgrmId = 30;
drvObj.Telegrams.CreateTelegram(tgrmId);

// Get and set safety telegram attributes
uint watchDogTime =
(uint)safetyTgrm.GetAttribute("Failsafe_FMonitoringtime");

safetyTgrm.SetAttribute("Failsafe_FMonitoringtime", 300);

const int newSafetyTelegramNumber= 902;
if (safetyTgrm.CanChangeTelegram(newSafetyTelegramId)) {
safetyTgrm.TelegramId = newSafetyTelegramId; }
```

The following example indicates how you can remove a telegram.

Remove telegram

```
using Siemens.Engineering.MC.DriveConfiguration;
//Delete telegram
const int tgrmId = 136;
drvObj.Telegrams.DeleteTelegram(tgrmId);
```

Inserting and deleting safety telegrams

The following example shows how to insert a Safety telegram. You require a drive object for access.

Inserting a Safety telegram and accessing telegram attributes

```
using Siemens.Engineering.MC.DriveConfiguration;
```

Inserting a Safety telegram and accessing telegram attributes

```

TelegramComposition telegrams = drvObj.Telegrams;

//Add safety telegram
const int tgrmId = 30;
drvObj.Telegrams.Create(tgrmId);

// Get and set safety telegram attributes
uint Failsafe_FDestinationAddress =
(uint)safetyTelegram.GetAttribute("Failsafe_FDestinationAddress");
uint Failsafe_FSourceAddress =
(uint)safetyTelegram.GetAttribute("Failsafe_FSourceAddress");
uint Failsafe_FIODBNumber = (uint)safetyTelegram.GetAttribute("Failsafe_FIODBNumber");
string Failsafe_FIODBName = safetyTelegram.GetAttribute("Failsafe_FIODBName").ToString();
uint Failsafe_FMonitoringtime =
(uint)safetyTelegram.GetAttribute("Failsafe_FMonitoringtime");
uint Failsafe_ManualAssignmentFIODBNumber =
(uint)safetyTelegram.GetAttribute("Failsafe_ManualAssignmentFIODBNumber");
bool Failsafe_ManualAssignmentFMonitoringtime =
(bool)safetyTelegram.GetAttribute("Failsafe_ManualAssignmentFMonitoringtime");

// Set safety telegram attributes
safetyTelegram.SetAttribute("Failsafe_ManualAssignmentFIODBNumber", 1);
safetyTelegram.SetAttribute("Failsafe_ManualAssignmentFMonitoringtime", true);
safetyTelegram.SetAttribute("Failsafe_FIODBNumber", 40000);
safetyTelegram.SetAttribute("Failsafe_FMonitoringtime", 200);
safetyTelegram.SetAttribute("Failsafe_FDestinationAddress", 15);

const int newSafetyTelegramId= 900;
if (safetyTgrm.CanChangeTelegram(newSafetyTelegramId)) {
safetyTgrm.TelegramId = newSafetyTelegramId; }

```

The following example shows how to delete a safety telegram.

Deleting a safety telegram

```

using Siemens.Engineering.MC.DriveConfiguration;
//Remove Safety telegram
drvObj.Telegrams.DeleteTelegram(TelegramType.SafetyTelegram);

```

Extending telegrams

The following example shows how to insert an extension and change the size of a standard telegram. You require a drive object for access.

Inserting an extension and changing the size of a standard telegram

```

using Siemens.Engineering.MC.DriveConfiguration;

```

Inserting an extension and changing the size of a standard telegram

```

TelegramComposition telegrams = drvObj.Telegrams;
Telegram telegram = telegrams.Find(TelegramType.MainTelegram);

Console.WriteLine("The Cu has the telegram: " + telegram.TelegramId);
Console.WriteLine("The Setpoint channel-specific size of the telegram is: "
+ telegram.GetOutputSize());

foreach (var address in telegram.Addresses)
{
    if (address.IoType == AddressIoType.Output)
    {
        Console.WriteLine("The Setpoint channel-specific IO start address of
the telegram on the connected PLC is: " + address.StartAddress);
    }
    else if (address.IoType == AddressIoType.Input)
    {
        Console.WriteLine("The Actual value channel-specific IO start address
of the telegram on the connected PLC is: " + address.StartAddress);
    }
}

// Create an additional telegram
if (drvObj.Telegrams.CreateAdditionalTelegram(2, 4))
{
    drvObj.Telegrams.CreateAdditionalTelegram(2, 4);
}

// Add a 3 word extension to the main telegram
Telegram mainTelegram == drvObj.Telegrams.Find(TelegramType.MainTelegram);
Int32 newSize = mainTelegram.GetSize(AddressIoType.Input) + 3;
if (mainTelegram.CanChangeSize(AddressIoType.Input, newSize, true))
{
    mainTelegram.ChangeSize(AddressIoType.Input, newSize, true)
}

```

7.27.6.10 Examples for Namespace Siemens.Engineering.MC.Drives

Preparing telegrams

The drive communication of a SINUMERIK NCU is realized using telegrams via the SINAMICS Integrated subcomponent and, if applicable, via additionally connected NX modules.

Note

The SINUMERIK NCU and a SINAMICS Integrated are located at the same level in the TIA Portal Openness object model, and appear under "DeviceComposition" as two different devices.

Use "DriveObjectContainer" to configure a telegram. "DriveObjectContainer" is a service of the drive object (device element) of the actual header module (device element).

To start the "DriveObjectContainer" service, navigate to SINAMICS Integrated or to the header module of the NX module. The hierarchy between devices and device elements is identical for SINAMICS Integrated and NX modules.

The following example shows how to find "DriveObjectContainer" via the "Header module" property:

Finding DriveObjectContainer via header module

```
foreach (Device device in project.Devices)
{
    foreach (DeviceItem deviceItem in device.DeviceItems)
    {
        if (deviceItem.Classification == DeviceItemClassifications.HM)
        {
            var driveObjectContainer = deviceItem.GetService<DriveObjectContainer>();
            // do something
        }
    }
}
```

The SINUMERIK NCU contains a SINAMICS Integrated proxy object with references to SINAMICS Integrated.

To access a SINAMICS Integrated device or an NX module, from the SINUMERIK NCU navigate to the DP Integrated interface via NCK. Then determine the PROFIBUS master system and navigate to the connected slave.

Inserting and removing telegrams

The following example indicates how you can insert a telegram. You require a drive object for access.

Inserting telegrams and accessing telegram attributes

```
using Siemens.Engineering.MC.Drives;
```

Inserting telegrams and accessing telegram attributes

```

TelegramComposition telegrams = drvObj.Telegrams;

//Add telegram
const int tgrmNumber = 136;
drvObj.Telegrams.InsertTelegram(tgrmNumber);

//Find telegram
Telegram telegram = drvObj.Telegrams.Find(TelegramType.MainTelegram);

//Add safety telegram
const int tgrmNumber = 30;
drvObj.Telegrams.InsertSafetyTelegram(tgrmNumber);

//Find safety telegram
Telegram safetyTgrm = drvObj.Telegrams.Find(TelegramType.SafetyTelegram);

// Get and set safety telegram attributes
uint watchDogTime =
(uint)safetyTgrm.GetAttribute("Failsafe_FMonitoringtime");

safetyTgrm.SetAttribute("Failsafe_FMonitoringtime", 300);

const int newSafetyTelegramNumber= 902;
if (safetyTgrm.CanChangeTelegram(newSafetyTelegramNumber)) {
safetyTgrm.TelegramNumber = newSafetyTelegramNumber; }

//Add supplementary telegram
const int tgrmNumber = 701;
drvObj.Telegrams.InsertSupplementaryTelegram(tgrmNumber);

Telegram telegram =
drvObj.Telegrams.Find(TelegramType.SupplementaryTelegram);

```

The following example indicates how you can remove a telegram.

Remove telegram

```

using Siemens.Engineering.MC.Drives;
//Remove safety telegram
drvObj.Telegrams.EraseTelegram(TelegramType.SafetyTelegram);

//Remove supplementary telegram
drvObj.Telegrams.EraseTelegram(TelegramType.SupplementaryTelegram);

```

Note

You can change, but not delete, a main telegram (MainTelegram).

Working with Safety telegrams

The following example shows how to insert a Safety telegram. You require a drive object for access.

Inserting a Safety telegram and accessing telegram attributes

```
using Siemens.Engineering.MC.Drives;
TelegramComposition telegrams = drvObj.Telegrams;

//Add safety telegram
const int tgrmNumber = 30;
drvObj.Telegrams.InsertSafetyTelegram(tgrmNumber);

//Find safety telegram
Telegram safetyTgrm = drvObj.Telegrams.Find(TelegramType.SafetyTelegram);

// Get and set safety telegram attributes
uint Failsafe_FDestinationAddress =
(uint)safetyTelegram.GetAttribute("Failsafe_FDestinationAddress");
uint Failsafe_FSourceAddress =
(uint)safetyTelegram.GetAttribute("Failsafe_FSourceAddress");
uint Failsafe_FIODBNumber = (uint)safetyTelegram.GetAttribute("Failsafe_FIODBNumber");
string Failsafe_FIODBNName = safetyTelegram.GetAttribute("Failsafe_FIODBNName").ToString();
uint Failsafe_FMonitoringtime =
(uint)safetyTelegram.GetAttribute("Failsafe_FMonitoringtime");
uint Failsafe_ManualAssignmentFIODBNumber =
(uint)safetyTelegram.GetAttribute("Failsafe_ManualAssignmentFIODBNumber");
bool Failsafe_ManualAssignmentFMonitoringtime =
(bool)safetyTelegram.GetAttribute("Failsafe_ManualAssignmentFMonitoringtime");

// Set safety telegram attributes
safetyTelegram.SetAttribute("Failsafe_ManualAssignmentFIODBNumber", 1);
safetyTelegram.SetAttribute("Failsafe_ManualAssignmentFMonitoringtime", true);
safetyTelegram.SetAttribute("Failsafe_FIODBNumber", 40000);
safetyTelegram.SetAttribute("Failsafe_FMonitoringtime", 200);
safetyTelegram.SetAttribute("Failsafe_FDestinationAddress", 15);

const int newSafetyTelegramNumber= 900;
if (safetyTgrm.CanChangeTelegram(newSafetyTelegramNumber)) {
safetyTgrm.TelegramNumber = newSafetyTelegramNumber; }
```

The following example shows how to delete a safety telegram.

Deleting a safety telegram

```
using Siemens.Engineering.MC.Drives;
//Remove Safety telegram
drvObj.Telegrams.EraseTelegram(TelegramType.SafetyTelegram);
```

Extending telegrams

The following example shows how to insert an extension and change the size of a standard telegram. You require a drive object for access.

Inserting an extension and changing the size of a standard telegram

```
using Siemens.Engineering.MC.Drives;
```

Inserting an extension and changing the size of a standard telegram

```

TelegramComposition telegrams = drvObj.Telegrams;
Telegram telegram = telegrams.Find(TelegramType.MainTelegram);

Console.WriteLine("The Cu has the telegram: " + telegram.TelegramNumber);
Console.WriteLine("The Setpoint channel-specific size of the telegram is: "
+ telegram.GetOutputSize());

foreach (var address in telegram.Addresses)
{
    if (address.IoType == AddressIoType.Output)
    {
        Console.WriteLine("The Setpoint channel-specific IO start address of
the telegram on the connected PLC is: " + address.StartAddress);
    }
    else if (address.IoType == AddressIoType.Input)
    {
        Console.WriteLine("The Actual value channel-specific IO start address
of the telegram on the connected PLC is: " + address.StartAddress);
    }
}

// Add an additional telegram
if (drvObj.Telegrams.CanInsertAdditionalTelegram(2, 4))
{
    drvObj.Telegrams.InsertAdditionalTelegram(2, 4);
}

// Add a 3 word extension to the main telegram
Telegram mainTelegram == drvObj.Telegrams.Find(TelegramType.MainTelegram);
Int32 newSize = mainTelegram.GetSize(AddressIoType.Input) + 3;
if (mainTelegram.CanChangeSize(AddressIoType.Input, newSize, true))
{
    mainTelegram.ChangeSize(AddressIoType.Input, newSize, true)
}

```

7.28 Functions for Startdrive

7.28.1 Introduction

Using TIA Portal-Openness, you automate the engineering and control the TIA Portal using a program that you created yourself.

In this help document, you can find a lot of information and code examples for this program that you generate yourself. You can also generate and use your own programs for the TIA Portal "Startdrive" application.

Before you configure your own program for Startdrive from the subsequently listed code example, please carefully observe the general information relating to Openness, which you can find under the following keywords in this help document:

- Preconditions for using TIA Portal Openness
- Install TIA Portal Openness
- Access the TIA Portal
- TIA Portal Openness object model
- Programming steps

7.28.2 TypelIdentifier - identifier of the components

Each version of any Startdrive component comprises a unique number, which is called the TypelIdentifier . In the Openness program code, you can use these TypelIdentifiers in order to uniquely identify and name a component.

In Startdrive, the display of the TypelIdentifier is optional, and deactivated as default.

Activating the display of the TypelIdentifier in Startdrive

1. In the Startdrive project view, select menu "Options > Settings".
The "Settings" configuration area opens.
2. Select "Hardware configuration" in the secondary navigation.
3. Activate the option "Activate display of the Type Identifier for devices and modules".
The display of the TypelIdentifier is now active.

Reading out the TypelIdentifier in Startdrive

In a Startdrive project, the TypelIdentifier can be read out at the following locations when the display is active:

- For all components (in the inspector window):
- For Control Units (when creating a drive device)

To read out the TypelIdentifier for a component in the inspector window, proceed as follows:

1. In the device view of the Startdrive project, double-click on the required component.
The inspector window opens. The active component version is shown in the list.
2. The associated TypelIdentifiers are listed in the outer right-hand column.

Example: OrderNumber: 6SL3131-7TE23-6Axx
Copy this TypelIdentifier to your Openness application.

Proceed as follows to read out the TypelIdentifier for a Control Unit:

1. In the Startdrive project navigation, double click on "Add new device".
The dialog with the same name opens.
2. Select the required control module from the selection.
The TypelIdentifier (under the article number and firmware number) for the corresponding Control Unit is displayed to the right in the detailed display.
Example: OrderNumber:6SL3040-1MA01-0Axx/V5.2/S120
3. Copy this TypelIdentifier to your Openness application.

7.28.3 References

7.28.3.1 AddressComposition

AddressComposition

The AddressComposition class represents the address of a telegram.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **syntax** of the class:

```
public sealed class AddressComposition
```

The following table describes the **properties** of the class:

Name	Data type	Description
IoType	AddressIoType (Page 686)	Returns information on the type of the address.
Context	AddressContext (Page 686)	Returns information on the context of the address.
StartAddress	Int32	Returns the start address of the telegram or specifies it.
Length	Int32	Returns the length of the telegram.

See also

TelegramType (Page 699)

7.28.3.2 AddressContext

AddressContext

The Enum AddressContext contains information on the context of the address.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **syntax** of the class:

```
public enum AddressContext
```

The following table describes the **enum entries**:

Name	Description
AddressContext.None	No context has been found for the address
AddressContext.Device	Context is a device address
AddressContext.Head	Context is a head address

7.28.3.3 AddressIoType

AddressIoType

The Enum AddressIoType contains information on the type of the address.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **syntax** of the class:

```
public enum AddressIoType
```

The following table describes the **enum entries**:

Name	Description
AddressIoType.None	The IO type cannot be used
AddressIoType.Input	Type is an input address
AddressIoType.Output	Type is an output address
AddressIoType.Diagnosis	Type is a diagnosis address
AddressIoType.Substitute	Type is a substitute address

7.28.3.4 ConfigurationEntry

ConfigurationEntry

Class ConfigurationEntry is used to save parameter data, which can be determined from the ConfigurationEntryCompositions of a motor or encoder configuration.

The following table describes the **properties** of the class:

Name	Data type	Description
EnumValueList	IDictionary<it, string>	Returns a list with possible values of the enum parameter.
MaxValue	object	Maximum value of the ConfigurationEntry.
MinValue	object	Minimum value of the ConfigurationEntry.
Name	string	Name of the ConfigurationEntry.
Number	int	Numerical representation of the name for ConfigurationEntry.
Description	string	Description of the ConfigurationEntry.
Parent	IEngineeringObject	Engineering Object model-parent element of this object.
Unit	string	Unit of the ConfigurationEntry.
Value	object	Value of the ConfigurationEntry.

7.28.3.5 DriveDomainFunctions

DriveDomainFunctions

Class DriveDomainFunctions is used to restore the factory settings or the backup of the RAM content to the ROM.

It can only be applied to an OnlineDriveFunctionInterface object.

For G120 drives, the DriveDomainFunctions object can only be accessed if the Power Module is connected to the device. Otherwise, null or an exception is returned.

The following table describes the **methods** of the class:

Name	Description
PerformFactoryReset	This method is responsible for restoring the factory settings.
PerformRAMtoROMCopyAllDriveObject	The date of all drive objects is written from the RAM to the memory card/hard disk.

The following table describes the **properties** of the class:

Name	Data type	Description
Parent	IEngineeringObject	Engineering Object model-parent element of this object.

7.28.3.6 DriveObject

DriveObject

The `DriveObject` class allows access to the drive object. Access to the drive parameters or the telegram is possible via the drive object, for example.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **syntax** of the class:

```
public sealed class DriveObject
```

The following table describes the **properties** of the class:

Name	Data type	Description
Parameters	DriveParameter-Composition (Page 691)	Returns a list with the available parameters of the drive object.
Telegrams	TelegramCompo-sition (Page 698)	Returns a list with the available telegrams of the drive object. The list can be changed with class <code>TelegramComposition</code> .

See also

Determining a drive object (Page 703)

7.28.3.7 DriveObjectActivation

DriveObjectActivation

Class `DriveObjectActivation` is used to activate the modules or determine the module status. It can be applied to `DriveObjectFunctions` from `DriveFunctionInterface` or `OnlineDriveFunctionInterface`.

The following table describes the **methods** of the class:

Name	Description
ChangeActivationState (DriveObjectActivationState)	Changes the activation status of the drive object. Returns false if the operation cannot be completed. Possible status values: <ul style="list-style-type: none">• Deactivate• Activate• DeactivateAndNotPresent

The following table describes the **properties** of the class:

Name	Data type	Description
ActivationState	DriveObjectActivationState	Returns the actual status value.
IsActive	Boolean	For an active drive object, returns true.

7.28.3.8 DriveObjectContainer

DriveObjectContainer

The `DriveObjectContainer` is a service of the drive object (`DeviceItem`) for the current device (`Device`).

The following table describes the **navigators** of the `DriveObjectContainer`:

Name	Data type	Description
DriveObjects	DriveObjectComposition (Page 691)	Returns a list with the available drive objects. The drive objects allow access to the drive parameters and telegrams.

7.28.3.9 DriveObjectTypeHandler

DriveObjectTypeHandler

Class `DriveObjectTypeHandler` is used to switch over the drive object type for every drive object and to determine the drive object type - as well as all possible drive object types of the actual drive object. It can only be applied to `DriveFunctionInterface`.

The following table describes the **methods** of the class:

Name	Description
<code>ChangeDriveObjectType((target)DriveObjectType)</code>	Changes the actual type of the drive object to a new type that can be selected. Returns false if the operation cannot be completed.

The following table describes the **properties** of the class:

Name	Data type	Description
<code>PossibleDriveObjectTypes</code>	<code>DriveObjectTypeComposition</code>	The use of <code>targetDriveObjectType</code> results in an exception on the actual drive object.
<code>CurrentDriveObjectType</code>	<code>DriveObjectType</code>	Indicates the currently assigned drive object type.

7.28.3.10 DriveParameter

DriveParameter

The `DriveParameter` class allows access to a drive parameter. Not all drive parameters are available for access via Openness.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **syntax** of the class:

```
public sealed class DriveParameter
```

The following table describes the **properties** of the class:

Name	Data type	Description
ArrayIndex	Int32	Returns the index of an array parameter. Value range: 0-7FFF The index is -1 for parameters without array Example <code>p108[4].15</code> <code>par.ArrayIndex</code> produces 4
ArrayLength	Int32	Returns the number of array elements. The index is 0 for parameters without an array
Bits	DriveParameter-Composition (Page 691)	Returns an <code>DriveParameter</code> object for one bit of the parameter. In this way, for example, the value or the name of the bit parameter can be read from a bit parameter. Example <code>DriveParameter param133 = cu.Parameters.Find(133, 0);</code> <code>DriveParameter param133Bit1 = param133.Bits[1];</code> <code>String paramName = param133Bit1.Name;</code>
EnumValueList	IDictionary<int, string>	Returns a list with possible values of the enum parameter. For example <1, [1] Quick commissioning> null, if the parameter is not a parameter of the <code>Enum</code> type.
MaxValue	Object	Returns the maximum value for the currently selected unit.
MinValue	Object	Returns the minimum value for the currently selected unit.
Name	string	Returns the name of the parameter. E.g. "p108[0].2"
ParameterText	string	Returns the text of the short description for the parameter.
Number	Int32	Returns the number of the parameter. Example <code>p108[0].2</code> The return value is 108

Name	Data type	Description
Unit	string	Returns the unit of the parameter as text.
Value	Object	<p>Returns the offline/online value of the parameter or writes a value onto the parameter.</p> <p>If the event of write errors, an <code>EngineeringTargetInvocationException</code> is triggered.</p> <p>Examples</p> <ul style="list-style-type: none"> • <code>P2080Bit6.Value = 0;</code> • <code>P2080Bit6.Value = cu.Parameters.Find("r19");</code> <p>BICO source</p> <p>The parameters of a BICO source can only be read</p> <p>BICO signal sinks</p> <p>Possible values are 0, 1 or a <code>DriveParameter</code> object.</p> <p>A <code>DriveParameter</code> object is returned when the BICO signal sink is connected to a different parameter.</p> <p>See also example Reading and writing BICO parameters (Page 704).</p>

See also

[Reading and writing parameters \(Page 720\)](#)

7.28.3.11 DriveParameterComposition

DriveParameterComposition

The `DriveParameterComposition` class allows access to parameters of the drive. Not all drive parameters are approved for access via Openness.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **syntax** of the class:

```
public sealed class DriveParameterComposition
```

The following table describes the **methods** of the class:

Name	Description
Find(string)	Returns the DriveParameter (Page 690) object for which a search is being made via the name. null if the parameter is not found Example <code>Find("P108[1]");</code>
Find(UInt16, Int32)	Returns the DriveParameter (Page 690) object for which a search is being made via the parameter index and array index. null if the parameter is not found Examples <ul style="list-style-type: none"> • cu.Find(108, 1); • cu.Find(51, -1);
WriteParameters(IEnumerable<string>, IEnumerable<string>, bool)	Writes values in parameters. With the ignoreErrors = true setting, an attempt is made to write all values in the event of an error and an EngineeringTargetInvocationException is triggered at the end. For SINAMICS G drives, parameter values can only be written with a configured Power Module (PM). Example <code>List<string> names = new List<string>(); List<string> values = new List<string>(); names.add("p300[0]"); values.add("17"); names.add("p5391[0]"); values.add("20"); cu.WriteParameters(names, values, true);</code>

7.28.3.12 EncoderConfiguration

EncoderConfiguration

Class `EncoderConfiguration` saves data of non-Siemens encoders.

- The user must populate the `ConfigurationEntryComposition` object.
- Object `RequiredConfigurationEntries` must also be populated.

Namespace: Siemens.Engineering.MC.Drives

Siemens.Engineering.MC.Drives.DFI

Siemens.Engineering.MC.Drives.Enum

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **properties** of the class:

Name	Data type	Description
RequiredConfigurationEntries	ConfigurationEntryComposition	Accessible configuration entries of the motor configuration.
Parent	IEngineeringObject	Engineering Object model-parent element of this object.

7.28.3.13 HardwareProjection

Class `HardwareProjection` is responsible for commissioning the motor and the encoder. The object can be found for `DriveFunctionInterface` and `OnlineDriveFunctionInterface`.

For G120 drives, motors and encoders can be configured both online and offline. On the other hand, for S120 drives, configuration is only possible offline.

For G120 drive devices, the `HardwareProjection` object can only be accessed if the Power Module is inserted in the drive device. Otherwise, when calling functions `HardwareProjection`, a null or an exception is returned.

For an offline configuration, use the hardware configuration of the `DriveFunctionInterface`. For an online configuration, use the hardware configuration of the `OnlineDriveFunctionInterface`.

Namespace: Siemens.Engineering.MC.Drives
 Siemens.Engineering.MC.Drives.DFI
 Siemens.Engineering.MC.Drives.Enums

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **methods** of the class:

Name	Description
<code>SetMotorType(MotorType type, ushort driveDataSet)</code>	Sets the motor type at the Control Unit (only for G120).
<code>GetCurrentMotorConfiguration(ushort driveDataSet)</code>	Depending on the data set number of the drive, determines the currently existing configuration area.
<code>ProjectMotorConfiguration(MotorConfiguration motConfig, ushort driveDataSet)</code>	Configures the motor configuration of a drive device depending on the data set number of the drive.
<code>SetEncoder(EncoderType type, EncoderInterface interfaceType, AbsoluteIncrementalFlag absIncFlag, RotaryLinearFlag rotLinFlag, ushort encDataSet)</code>	Sets the encoder at the Control Unit (only for G120).
<code>GetCurrentEncoderConfiguration(ushort encDataSet)</code>	Depending on the data set number of the encoder, determines the currently existing configuration area.
<code>ProjectEncoderConfiguration(EncoderConfiguration encConfig, ushort encDataSet)</code>	Configures the motor configuration of a drive device depending on the data set number of the encoder.

The following table describes the **properties** of the class:

Name	Data type	Description
Parent	IEngineeringObject	Engineering Object model-parent element of this object.

7.28.3.14 MotorConfiguration

MotorConfiguration

Class `MotorConfiguration` is responsible for commissioning motors and encoders.

Namespace: Siemens.Engineering.MC.Drives
 Siemens.Engineering.MC.Drives.DFI
 Siemens.Engineering.MC.Drives.Enums
Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Response for Siemens motors:

Action `SetMotortype` must be called to set the motor type. This action comprises the Enum `MotorType` and a number, which represents the `DriveDatasetNumber`.

The following table describes the enums:

Enum name	Description
NoMotor	0: No motor
InductionMotor	1: Induction motor
SynchronousMotor	2: Synchronous motor
NoCodeNumber1LE1InductionMotor	10: 1LE1 induction motor (not a code number)
NoCodeNumber1LG6InductionMotor	13: 1LG6 induction motor (not a code number)
NoCodeNumber1xx1SIMOTICSFDInductionMotor	14: 1xx1 SIMOTICS FD induction motor (not a code number)
NoCodeNumber1LA7InductionMotorNoCodeNumber	17: 1LA7 induction motor (not a code number)
MotorSeriesNumber1LA81PQ8StandardInduction	18: 1LA8 / 1PQ8 standard induction motor series
NoCodeNumber1LA9InductionMotor	19: 1LA9 induction motor (not a code number)

Response for non-Siemens motors:

Class `MotorConfiguration` is used to save data of non-Siemens motors. It contains 2 objects `ConfigurationEntryComposition`, which, when commissioning, must be populated with the corresponding motor data. Object `RequiredConfigurationEntries` must also be populated. Object `OptionalConfigurationEntries` does not have to be populated.

The following table describes the **properties** of the class:

Name	Data type	Description
RequiredConfigurationEntries	ConfigurationEntryComposition	Accessible configuration entries of this motor configuration.
OptionalConfigurationEntries	ConfigurationEntryComposition	Accessible configuration entries of this motor configuration.
Parent	IEngineeringObject	Engineering Object model-parent element of this object.

7.28.3.15 OnlineDriveObject

OnlineDriveObject

The `OnlineDriveObject` class allows online access to the drive object. Drive parameters can be accessed via the drive object.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **syntax** of the class:

```
public sealed class OnlineDriveObject
```

The following table describes the **properties** of the class:

Name	Data type	Description
Parameters	DriveParameter-Composition (Page 691)	Returns a list with the available parameters of the online drive object. null, if the mode is "Offline". In offline mode, an exception is triggered when a method is called or for a write access to a parameter.

See also

Determining a drive object (Page 703)

Reading and writing parameters (Page 720)

Reading and writing parameters online (Page 722)

7.28.3.16 OnlineDriveObjectContainer

OnlineDriveObjectContainer

The `OnlineDriveObjectContainer` is a service of the drive object (`DeviceItem`) for the current device (`Device`).

The following table describes the **navigators** of the `OnlineDriveObjectContainer`:

Name	Data type	Description
<code>OnlineDriveObjects</code>	<code>OnlineDriveObjectComposition</code>	Returns a list with the available online drive objects (<code>OnlineDriveObject</code> (Page 695)). The drive objects allow access to the drive parameters.

7.28.3.17 StartDriveDownloadCheckConfiguration

StartDriveDownloadCheckConfiguration

Class `StartDriveDownloadCheckConfiguration` is derived from the class `DownloadCheckConfiguration` and has the same properties.

Class `DownloadCheckConfiguration` is described in the standard Openness help.

The class provides the configuration settings via checkboxes from the user.

The following table describes the **properties** of the class:

Name	Data type	Description
<code>Checked</code>	<code>bool</code>	Returns the current setting of the configuration or activates/deactivates the configuration.

See also

[Download \(Page 705\)](#)

7.28.3.18 SafetyTelegram

SafetyTelegram

Class `SafetyTelegram` stands for the telegram of the drive object.

Exception `EngineeringTargetInvocationException` is displayed for errors in the write attributes.

Namespace: `Siemens.Engineering.MC.Drives`

Assembly: `Siemens.Engineering.MC.Drives` in `Siemens.Engineering.dll`

The following table describes the **properties** of the class:

Name	Data type	Description
<code>TelegramNumber</code>	<code>Int32</code>	Number of the main telegram. Free telegrams are designated with a value of 999.
<code>Type</code>	<code>TelegramType</code> (Page 699)	Telegram type that is returned as enum "TelegramType".

Name	Data type	Description
Addresses	AddressComposition (Page 685)	Composition of all addresses of a telegram.
PKW	Telegram (Page 697)	Composition of all PKW channels of the telegram. null - if the telegram does not have a PKW part.

7.28.3.19 Telegram

Telegram

Class `Telegram` allows access to the structure of a telegram from a drive object.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **syntax** of the class:

```
public sealed class Telegram
```

The following table describes the **properties** of the class:

Name	Data type	Description
TelegramNumber	Int32	Returns the number of the main telegram or specifies the number. A freely configurable telegram has the number 999.
Type	TelegramType (Page 699)	Returns the type of telegram as <code>Enum TelegramType</code> .
Addresses	AddressComposition (Page 685)	Returns an <code>AddressComposition</code> with information on the address.
PKW	Telegram	Returns channel PKW as <code>Telegram</code> . null if the property is not available A telegram with PKW is telegram 353, for example.

The following table describes the **methods** of the class:

Name	Description
<code>CanChangeTelegram(Int32)</code>	Returns <code>true</code> if the telegram can be changed to the parameterized standard type.
<code>GetSize(AddressIoType (Page 686))</code>	Returns the size of the inputs or outputs of the telegram.
<code>CanChangeSize(AddressIoType (Page 686), Int32, bool)</code>	Returns <code>true</code> if the size of the telegram can be changed as parameterized. Standard telegrams can only be enlarged. The retention of the previous telegram address is taken into account when the option is parameterized with <code>true</code> .
<code>ChangeSize(AddressIoType (Page 686), Int32, bool)</code>	Returns <code>true</code> if the size of the telegram could be changed as parameterized. The retention of the previous telegram address is taken into account when the option is parameterized with <code>true</code> .

7.28.3.20 TelegramComposition

TelegramComposition

The `TelegramComposition` class allows access to the telegrams of a drive object. The structure of a telegram can be read out via class `Telegram` (Page 697).

Note that referenced objects may become invalid through class `TelegramComposition`. For example, object `Telegram` (Page 697) becomes invalid after the telegram size is changed.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **syntax** of the class:

```
public sealed class TelegramComposition
```

The following table describes the **methods** of the class:

Name	Description
<code>CanInsertAdditionalTelegram(Int32, Int32)</code>	Returns <code>true</code> if an extension can be created in accordance with the parameterized sizes (input and output sizes).
<code>CanInsertTorqueTelegram(Int32, telegramNumber)</code>	When adding a new torque telegram with an already existing telegram number, checks whether this is possible.
<code>CanInsertSupplementaryTelegram(Int32)</code>	Returns <code>true</code> if a supplementary telegram can be created in accordance with the parameterized telegram number.
<code>EraseTelegram(TelegramType)</code>	Deletes a telegram with a known telegram type from the drive object. Returns <code>true</code> if the parameterized telegram could be deleted. If a torque telegram is used as type, then object "torqueTelegram" is deleted. If a Safety Integrated telegram is used as type, then object "safetyTelegram" is deleted. Standard telegrams cannot be deleted. In the event of an error, an <code>EngineeringTargetInvocationException</code> is initiated.
<code>Find(TelegramType)</code>	Returns the <code>Telegram</code> (Page 697) object if it could be found via the parameterized telegram type or Safety Integrated telegram type. <code>null</code> if the telegram is not found. If a torque telegram is used as type, then object <code>torqueTelegram</code> is returned, assuming that it exists. If a Safety Integrated telegram is used as type, then object <code>safetyTelegram</code> is returned, assuming that it exists. Example <code>Telegram telegram = telegrams.Find(TelegramType.MainTelegram);</code>

Name	Description
InsertAdditionalTelegram(Int32, Int32)	Creates an extension for the drive object in accordance with the parameterized sizes and returns <code>true</code> if the extension could be inserted. In the event of an error, <code>an EngineeringTargetInvocationException</code> is initiated.
InsertTorqueTelegram(Int32, telegramNumber)	Adds a new torque telegram with an already existing telegram number to a drive object.
InsertSafetyTelegram(Int32, telegramNumber)	Adds a Safety Integrated telegram with its specified telegram number to a drive object.
InsertSupplementaryTelegram(Int32)	Creates the supplementary telegram with the parameterized telegram number and returns <code>true</code> if the telegram could be inserted. In the event of an error, <code>an EngineeringTargetInvocationException</code> is initiated.

7.28.3.21 TelegramType

TelegramType

The Enum `TelegramType` contains predefined telegram types.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **syntax** of the class:

```
public enum TelegramType
```

The following table describes the **enum entries**:

Name	Description
MainTelegram	ID of the main telegram
SupplementaryTelegram	ID of the supplementary telegram
AdditionalTelegram	ID of an extension

7.28.3.22 TorqueTelegram

TorqueTelegram

Class `TorqueTelegram` stands for the telegram of the drive object.

`Exception EngineeringTargetInvocationException` is displayed for errors in the write attributes.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

The following table describes the **properties** of the class:

Name	Data type	Description
TelegramNumber	Int32	Telegram number.
Type	TelegramType (Page 699)	Telegram type that is returned as enum "TelegramType".
Addresses	AddressComposition (Page 685)	Composition of all addresses of a telegram.
PKW	Telegram (Page 697)	Composition of all PKW channels of the telegram. null - if the telegram does not have a PKW part.

7.28.4 Code examples

The following code examples describe the basic procedure for various applications. The code is not necessarily complete or compilable.

7.28.4.1 Determining the activation status

The following examples show how you can determine the activation status for S120 drives, either offline or online:

Determining the activation status of an S120 drive offline

```
using Siemens.Engineering.MC.Drives;
DriveFunctionInterface dfi = ...
DriveObjectActivation driveObjectActivation =
dfi.DriveObjectFunctions.DriveObjectActivation;
//driveObjectActivation can be null in case of the actual driveobject does not support
activation.

//change activation state
driveObjectActivation.ChangeActivationState(DriveObjectActivationState.Deactivate);

//get the activation state
DriveObjectActivationState activationState = driveObjectActivation.ActivationState;

//get the Is Active property
bool isActive = driveObjectActivation.IsActive;
```

Determining the activation status of an S120 drive online

```
using Siemens.Engineering.MC.Drives;
```

Determining the activation status of an S120 drive online

```
OnlineDriveFunctionInterface onlinedfi = ...
DriveObjectActivation driveObjectActivation = onlinedfi.DriveObjectActivation;
//driveObjectActivation can be null in case of the actual driveobject does not support
activation in online.

//change activation state
driveObjectActivation.ChangeActivationState(DriveObjectActivationState.Deactivate);

//get the activation state
DriveObjectActivationState activationState = driveObjectActivation.ActivationState;

//get the IsActive property
bool isActive = driveObjectActivation.IsActive;
```

7.28.4.2 Executing drive functions

The following examples show how you can simply access drive functions, either offline or online:

Executing drive functions offline

```
using Siemens.Engineering.MC.Drives;
DriveObject driveObject = ...
DriveFunctionInterface dfi = driveObject.GetService<DriveFunctionInterface>();

// dfi can be null in case of the actual driveobject does not support it.
```

Executing drive functions online

```
using Siemens.Engineering.MC.Drives;
OnlineDriveObject onlineDriveObject = ...
OnlineDriveFunctionInterface onlineDfi =
onlineDriveObject.GetService<OnlineDriveFunctionInterface>();

// onlineDfi can be null in case of the actual onlineDriveObject
// does not support it or if the device is offline.
```

7.28.4.3 Creating a drive unit

You can create a drive unit with method `CreateWithItem()` of collection `Devices`.

The drive units are specified via parameters of the method. The format of the parameters is described in the following.

Creating a G120 drive unit

Format of the parameters for a G120:

```
CreateWithItem(@"OrderNumber: mlfb / FirmwareVersion /",
"NameOfTheDevice", positionNumber)
```

The following example shows how to create a G120 drive unit.

Creating a G120 drive unit

```
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
Project tiaproject= portal.Projects.Open("..."); //The path of the project

Device s120Device =
tiaproject.Devices.CreateWithItem(@"OrderNumber:6SL3246-0BA22-1FA0/4.7.6/"
, "Device_0", null);
```

Creating a S120, S150, MV, G130, G150 drive unit

Format of the parameters for a S120, S150, MV, G130, G150:

```
CreateWithItem(@"OrderNumber: mlfb / FirmwareVersion /
AdditionalTypeIdentifier", "NameOfTheDevice", positionNumber)
```

Possible values for AdditionalTypeIdentifier:

- Empty string (e.g. for G120)
- S120
- S150
- MV
- G130
- G150

The following example shows how to create a S120 drive unit.

Creating an S120 drive unit

```
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
Project tiaproject= portal.Projects.Open("..."); //The path of the project

Device s120Device =
tiaproject.Devices.CreateWithItem(@"OrderNumber:6SL3040-1MA01-0Axx/V4.8/
S120", "Device_0", null);
```

7.28.4.4

Creating a drive component

You can create a drive component for a drive unit with the `PlugNew()` method of the `Device` object.

The following example shows how to create a drive component.

Creating a Motor Module

```
DeviceItem subModul = sdrDevice.PlugNew(@"OrderNumber:6SL3xxx-xxxxx-xxxx",
"MotorModul", 65535);
```

7.28.4.5 Determining a drive object

The following examples show how to determine drive objects offline and online.

Determining an offline drive object

```
using Siemens.Engineering.MC.Drives;
//G device
Project project = portal.Projects.Open("..."); //Destination folder to
open the project
DeviceItem item = project.Devices[0].Items[0].Items[0];
DriveObject driveObject =
item.GetService<DriveObjectContainer>().DriveObjects[0];

//S device
Project project = portal.Projects.Open("..."); //Destination folder to
open the project
DeviceItem item = project.Devices[0].Items[0];
DriveObject driveObject =
item.GetService<DriveObjectContainer>().DriveObjects[0];
```

Determining an online drive object

```
using Siemens.Engineering.MC.Drives;
//G device
Project project = portal.Projects.Open("..."); //Destination folder to
open the project
DeviceItem item = project.Devices[0].Items[0].Items[0];
OnlineDriveObject onlineDriveObject =
item.GetService<OnlineDriveObjectContainer>().OnlineDriveObjects[0];

//S device
Project project = portal.Projects.Open("..."); //Destination folder to
open the project
DeviceItem item = project.Devices[0].Items[0];
OnlineDriveObject onlineDriveObject =
item.GetService<OnlineDriveObjectContainer>().OnlineDriveObjects[0];
```

7.28.4.6 Determining the drive object type

The following examples shows how you can determine the actual drive object type – and the types that can be alternatively set.

Determining drive object types

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
```

Determining drive object types

```

DriveObject driveObject = ...
DriveFunctionInterface dfi = driveObject.GetService<DriveFunctionInterface>();
DriveObjectTypeHandler driveObjectTypeHandler =
dfi.DriveObjectFunctions.DriveObjectTypeHandler;

// dfi can be null in case of the actual driveobject does not support it.

// driveObjectTypeHandler can be null, if the actual driveObject does not support it.

// Get the possible drive object types on the drive object.
DriveObjectTypeComposition possibleDriveObjectTypes =
driveObjectTypeHandler.PossibleDriveObjectTypes;

// Get the current drive object type on the drive object.
DriveObjectType currentDriveObjectType = driveObjectTypeHandler.CurrentDriveObjectType;

//Call the ChangeDriveObjectType method with the current drive object type.
//The method parameter should be the target drive object type.
driveObjectTypeHandler.ChangeDriveObjectType(possibleDriveObjectTypes[0]);

```

7.28.4.7 Reading and writing BICO parameters

The following example shows how to read and write values of BICO parameters. You require a drive object for access.

Reading BICO parameters

```

using Siemens.Engineering.MC.Drives;
DriveParameter bicoSink= driveObject.Parameters.Find("p681");
if(bicoSink!=null)
{
    if(bicoSink.Value is DriveParameter)
    {
        DriveParamter bicoSourceValue = bicoSink.Value as DriveParameter;
        Console.WriteLine("The value of parameter " + bicoSink.Name + ":" + 
bicoSource.Name + " " + bicoSource.ParameterText);
    }
    else if (bicoSink.Value == null)
    {
        Console.WriteLine("Value contains an invalid connection or the source
parameter is not accessible via Openness");
    }
    else
    {
        Console.WriteLine("The value of parameter " + bicoSink.Name + ":" + 
bicoSink.Value.ToString());
    }
}

```

Writing BICO parameters

```
using Siemens.Engineering.MC.Drives;
```

Writing BICO parameters

```
DriveParameter bicoSource= driveObject.Parameters.Find("r19");
DriveParameter bicoSink = driveObject.Parameters.Find("p738");
if(bicoSource != null)
{
    try
    {
        bicoSink.Value = bicoSource;
    }
    catch(UserException ex)
    {
        Console.WriteLine("Write failure :" + ex.Message);
    }
}
```

7.28.4.8 Download

After starting the download, you must adapt and confirm the configuration settings. The configuration settings are provided as child objects of object `DownloadConfiguration` – and there are three different types:

- `StartDriveDownloadCheckConfiguration`
- `DownloadSelectionConfiguration`
- `DownloadPasswordConfiguration`

The following examples show the evaluation of the different types of configuration settings in the `PreDownload Delegate`.

Evaluation of the configuration settings after starting the download

```
using Siemens.Engineering.Download;
using Siemens.Engineering.Online;
```

Evaluation of the configuration settings after starting the download

```
static void PreDownload(DownloadConfiguration configuration)
{
    Console.WriteLine(configuration.Message);
    StartDriveDownloadCheckConfiguration sdcc = configuration as
StartDriveDownloadCheckConfiguration;
    if (sdcc != null)
    {
        sdcc.Checked = true;
        return;
    }

    DownloadPasswordConfiguration downloadPasswordConfiguration =
configuration as DownloadPasswordConfiguration;

    if (downloadPasswordConfiguration != null)
    {
        SecureString s = new SecureString();
        string passwordText = "password";
        foreach (var str in passwordText)
        {
            s.AppendChar(str);
        }

        downloadPasswordConfiguration.SetPassword(s);
        return;
    }

    DownloadSelectionConfiguration downloadSelectionConfiguration =
configuration as DownloadSelectionConfiguration;

    if (downloadSelectionConfiguration != null)
    {
        downloadSelectionConfiguration.SelectedIndex = 0;
        return;
    }
}
```

The following examples show how to download a project to the device.

Download to the S120 device

```
using Siemens.Engineering.Download;
using Siemens.Engineering.Online;
```

Download to the S120 device

```

try
{
    DeviceItem item = ... //device item of the CU (e.g. :
project.Devices[0].Items[0].Items[0])
    DownloadProvider downloadProvider = item.GetService<DownloadProvider>();

    DownloadConfigurationDelegate pre = PreDownload;
    DownloadConfigurationDelegate post = PostDownload;

    ConnectionConfiguration connConfiguration =
downloadProvider.Configuration;
    ConfigurationMode configurationMode = connConfiguration.Modes.Find("PN/
IE");
    ConfigurationPcInterface pcInterface = configurationMode.PcInterfaces[0];
    ConfigurationSubnet subnet = pcInterface.Subnets.Find(/*subnet name*/);
    IConfiguration configuration = subnet.Addresses.Find(/*IP address of the
device*/);
    downloadProvider.Download(configuration, pre, post,
DownloadOptions.Software);
}
catch (Exception e)
{
    Console.WriteLine(e.Message);
}

//configuration handling before download
static void PreDownload(DownloadConfiguration configuration)
{
    Console.WriteLine(configuration.Message);
    StartDriveDownloadCheckConfiguration dcc = configuration as
StartDriveDownloadCheckConfiguration ;
    if (dcc != null)
    {
        dcc.Checked = true;
    }
}

//configuration handling after download
static void PostDownload(DownloadConfiguration configuration)
{
    Console.WriteLine(configuration.Message);
}

```

7.28.4.9 Editing DRIVE-CLiQ connections

The following example shows how to edit DRIVE-CLiQ connections with Openness.

Editing DRIVE-CLiQ connections

```
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
```

Editing DRIVE-CLiQ connections

```

Project tiaproject = portal.Projects.Open(new FileInfo(@"C:\Users\testUser
\Documents\Automation\Project109\Project109.ap15")); //The path of the
project

Device device =
tiaproject.Devices.CreateWithItem(@"OrderNumber:6SL3040-1MA01-0Axx/V4.8/
S120", "Device_0", null); //Create the device

DeviceItem subModul = device.PlugNew(@"OrderNumber:6SL3x2x-1xxxx-xxxx",
"", 65535); //Plug a submodul (in this case : Motor modul)

DeviceItem cu = device.DeviceItems[1]; // The CU is always the 1
indexed in the DeviceItems of the Device

DeviceItem subModulDQInterface = subModul.DeviceItems[0]; //We need
the DQ interface from the submodul
DeviceItem cuDQInterface = cu.DeviceItems[3]; //This is the
DQ interface of the CU

NetworkPort subModulDQ =
((IEngineeringServiceProvider)subModulDQInterface.DeviceItems[0]).GetService<Net
workPort>(); //We need the DriveCliq port of the DQ interface from
the submodul
NetworkPort cuDQ =
((IEngineeringServiceProvider)cuDQInterface.DeviceItems[0]).GetService<Net
workPort>(); //We need the DriveCliq port of the DQ interface
from the CU

cuDQ.DisconnectFromPort(subModulDQ); //Delete the connection between the
two ports (automatically created when plugging a modul)
cuDQ.ConnectToPort(subModulDQ); //Create a new connection

```

7.28.4.10 Carrying out the first steps in Startdrive

The following example shows how you can localize or generate an active TIA Portal process.

Finding or generating a TIA Portal process

```

using Siemens.Engineering;
// Get the list of the running TIA Portal processes.
IList<TiaPortalProcess> procs = TiaPortal.GetProcesses();
TiaPortal portal;
// When there is at least one running TIA Portal, we will attach to the first from the list.
if (procs.Count != 0)
{
    portal = procs[0].Attach();
}
// When there is no running TIA Portal, we create one.
else
{
    portal = new TiaPortal(TiaPortalMode.WithUserInterface);
}

```

The following example shows how you can localize or generate a Startdrive project.

Finding or generating a Startdrive project

```
using Siemens.Engineering;
Project project;
// When the portal has one project, we save it in a variable.
if (portal.Projects.Count == 1)
{
    project = portal.Projects[0];
}
// When there is no existing project, we create one with a specific path, and the actual time
else
{
    project = portal.Projects.Create(
        new DirectoryInfo(@"C:\Projects\Project_" + DateTime.Now.Ticks),
        DateTime.Now.Ticks.ToString());
}
```

The following example shows how you can determine as to whether a specific Startdrive variant (package and version) is installed.

Determining whether the required Startdrive version is installed

```
using Siemens.Engineering;
if (tiaProcess.InstalledSoftware.Any(sw => sw.Name.Equals("SINAMICS Startdrive Advanced") && sw.Version.Equals("V15"))) { Console.WriteLine("Startdrive is available");}
// "V15" is the current startdrive version started at December 2017.
// "V15.1" will be the current startdrive version beginning with December 2018.
// "SINAMICS Startdrive Basic" and "SINAMICS Startdrive Advanced" are the 2 possible
startdrive function packages.
```

7.28.4.11 Defining the encoder type

The following example shows how you can set the encoder type or the encoder data set number offline.

Setting the encoder type and/or the encoder data set number offline via the hardware configuration

```
using Siemens.Engineering.MC.Drives
using Siemens.Engineering.MC.Drives.DFI
using Siemens.Engineering.MC.Drives.Enums;
```

Setting the encoder type and/or the encoder data set number offline via the hardware configuration

```

DeviceItem cuDeviceItem = m_Device.DeviceItems[1];
DriveObject cuDriveObject =
cuDeviceItem.GetService<DriveObjectContainer>().DriveObjects[0];

DriveFunctionInterface cuDriveFunctionInterface =
cuDriveObject.GetService<DriveFunctionInterface>();
HardwareProjection hardwareProjection = cuDriveFunctionInterface.HardwareProjection;

// To enable the setting of an EncoderType, the value of p96 should
// be set to 0 (Application class == [0] Expert) if it is present on
// the current G drive.
DriveParameter p96 = cuDriveObject.Parameters.Find("p96");
if (p96 != null)
{
    p96.Value = 0;
}

// Setting Encoder 1 on the drive.
// In case of encoders, we have to set several enums to define an
// encoder type. These enums are: EncoderInterface, EncoderType,
// AbsoluteIncrementalFlag, and RotaryLinearFlag.

// There can be a problem, if the given enum combination is not valid.
// In that case, it has to give back a feedback.

hardwareProjection.SetEncoder(
    EncoderInterface.Terminal,
    EncoderType.HTLTTL,
    AbsoluteIncrementalFlag.Incremental,
    RotaryLinearFlag.Rotary,
    1);

// It is possible to set 2 encoders to a motor, one with encoderNumber == 1
// and the other with encoderNumber == 2

```

The following example shows how you can set the encoder type or the encoder data set number using an online connection.

Setting the encoder type and/or the encoder data set number online via the hardware configuration

```

using Siemens.Engineering.MC.Drives
using Siemens.Engineering.MC.Drives.DFI
using Siemens.Engineering.MC.Drives.Enums;

```

Setting the encoder type and/or the encoder data set number online via the hardware configuration

```
DeviceItem cuDeviceItem = m_Device.DeviceItems[1];
OnlineDriveObject cuOnlineDriveObject =
cuDeviceItem.GetService<OnlineDriveObjectContainer>().OnlineDriveObjects[0];
OnlineDriveFunctionInterface cuDriveFunctionInterface =
cuOnlineDriveObject.GetService<OnlineDriveFunctionInterface>();
HardwareProjection hardwareProjection = cuDriveFunctionInterface.HardwareProjection;

// To enable the setting of an Encoder, the value of p96 should
// be set to 0 (Application class == [0] Expert) if it is present
// the current G drive.
DriveParameter p96 = cuOnlineDriveObject.Parameters.Find("p96");
if (p96 != null)
{
    p96.Value = 0;
}

// Setting Encoder 1 on the drive.
// In case of encoders we have to set several enums to define an
// encoder type. These enums are: EncoderInterface, EncoderType,
// AbsoluteIncrementalFlag, and RotaryLinearFlag.

// There can be a problem, if the given enum combination is not valid.
// In that case, it has to give back a feedback.

hardwareProjection.SetEncoder(
    EncoderInterface.Terminal,
    EncoderType.HTLTTL,
    AbsoluteIncrementalFlag.Incremental,
    RotaryLinearFlag.Rotary,
    1);

// It is possible to set 2 encoders to a motor, one with encoderNumber == 1
// and the other with encoderNumber == 2
```

The following example shows how you can read out the actual encoder configuration from the drive.

Reading out the encoder configuration

```
DeviceItem cuDeviceItem = m_Device.DeviceItems[1];
DriveObject cuDriveObject =
cuDeviceItem.GetService<DriveObjectContainer>().DriveObjects[0];
DriveFunctionInterface cuDriveFunctionInterface =
cuDriveObject.GetService<DriveFunctionInterface>();

HardwareProjection encoderProjection = cuDriveFunctionInterface.HardwareProjection;

// Before setting an encoder, p96 should be 0
// (See section "Projecting EncoderConfiguration")

encoderProjection.SetEncoder(
    EncoderInterface.Terminal,
    EncoderType.HTLTTL,
    AbsoluteIncrementalFlag.Incremental,
    RotaryLinearFlag.Rotary,
    1);

EncoderConfiguration encoderConfiguration =
encoderProjection.GetCurrentEncoderConfiguration(1);
```

The following example shows how you can configure the actual encoder configuration offline.

Configuring an encoder offline

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;
```

Configuring an encoder offline

```
// Project encoder configuration in Offline state
DeviceItem cuDeviceItem = m_Device.DeviceItems[1];

DriveObject cuDriveObject =
cuDeviceItem.GetService<DriveObjectContainer>().DriveObjects[0];
DriveFunctionInterface cuDriveFunctionInterface =
cuDriveObject.GetService<DriveFunctionInterface>();
HardwareProjection hardwareProjection = cuDriveFunctionInterface.HardwareProjection;

// Before setting an encoder type, p96 should be 0
// (See section "Projecting EncoderConfiguration")

// To Project an EncoderConfiguration, first set e.g. Encoder 1 with .SetEncoder(...).

hardwareProjection.SetEncoder(
    EncoderInterface.Terminal,
    EncoderType.HTLTTL,
    AbsoluteIncrementalFlag.Incremental,
    RotaryLinearFlag.Rotary,
    1);

// Get the current configuration of Encoder 1.
EncoderConfiguration config = hardwareProjection.GetCurrentEncoderConfiguration(1);

// Fill out Encoder 1's configuration values.
config.RequiredConfigurationEntries.ToList().ForEach(ce =>
{
    switch (ce.Name)
    {
        case "p405.0":
            ce.Value = 0;
            break;
        case "p405.1":
            ce.Value = 1;
            break;
        case "p408":
            ce.Value = 1024;
            break;
        case "p425":
            ce.Value = 2048;
            break;
        default:
            break;
    }
});

// Project the Encoder 1 configuration to the device.
bool result =
cuDriveFunctionInterface.HardwareProjection.ProjectEncoderConfiguration(config, 1);
```

The following example shows how you can configure the actual encoder configuration online.

Configuring an encoder online

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;
```

Configuring an encoder online

```

// Project encoder configuration in Online state
DeviceItem cuDeviceItem = m_Device.DeviceItems[1];
DeviceItem cuDeviceItemForOnline = m_Device.DeviceItems[0];
//You need the rack for going online on G120 Device, which is .DeviceItems[0]

// GoOnline...
// To use these function in Online, you have to use the OnlineDriveFunctionInterface
OnlineDriveObject onlineDriveObject =
cuDeviceItem.GetService<OnlineDriveObjectContainer>().OnlineDriveObjects[0];
OnlineDriveFunctionInterface onlineDfi =
onlineDriveObject.GetService<OnlineDriveFunctionInterface>();
HardwareProjection hardwareProjection = onlineDfi.HardwareProjection;

// Before setting an encoder, p96 should be 0
// (See section "Projecting EncoderConfiguration")

// To Project an EncoderConfiguration, first set e.g. Encoder 1 with .SetEncoder(...).

hardwareProjection.SetEncoder(
    EncoderInterface.Terminal,
    EncoderType.HTLTTL,
    AbsoluteIncrementalFlag.Incremental,
    RotaryLinearFlag.Rotary,
    1);

// Get the current configuration of Encoder 1.
EncoderConfiguration config = hardwareProjection.GetCurrentEncoderConfiguration(1);

// Fill out Encoder 1's configuration values.
config.RequiredConfigurationEntries.ToList().ForEach(ce =>
{
    switch (ce.Name)
    {
        case "p405.0":
            ce.Value = 0;
            break;
        case "p405.1":
            ce.Value = 1;
            break;
        case "p408":
            ce.Value = 1024;
            break;
        case "p425":
            ce.Value = 2048;
            break;
        default:
            break;
    }
});

// Project the Encoder 1 configuration to the device.
bool result = onlineDfi.HardwareProjection.ProjectEncoderConfiguration(config, 1);

```

The following example shows how you can write the configuration entry to the console.

Write out configuration entry

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
MotorConfiguration motorConfig = hardwareProjection.GetCurrentMotorConfiguration(0);

foreach (var configurationEntry in motorConfig.RequiredConfigurationEntries)
{
    Console.WriteLine(configurationEntry.Name);
}
EncoderConfiguration encConfig = hardwareProjection.GetCurrentEncoderConfiguration(1);

foreach (var configurationEntry in encConfig.RequiredConfigurationEntries)
{
    Console.WriteLine(configurationEntry.Name);
}
```

7.28.4.12 Configuring devices

The following example shows how you can configure S120 and G120 drive devices offline.

Configuring S120 and/or G120 drive devices offline

```
using Siemens.Engineering.MC.Drives
using Siemens.Engineering.MC.Drives.DFI
using Siemens.Engineering.MC.Drives.Enums;

DriveObject driveObject = ...
DriveFunctionInterface dfi = driveObject.GetService<DriveFunctionInterface>();
HardwareProjection hardwareProjection = dfi.HardwareProjection;

// dfi can be null in case of the actual driveobject does not support it.
// hardwareProjection can be null, if the actual driveObject does not support it.
// For example: On G120 drives, you have to use the CU as driveobject. On S120
// drives, you have to use the MotorModul as driveObject.
```

The following example shows how you can configure G120 drive devices online.

Configuring G120 drive devices online

```
using Siemens.Engineering.MC.Drives
using Siemens.Engineering.MC.Drives.DFI
using Siemens.Engineering.MC.Drives.Enums;

OnlineDriveObject onlineDriveObject = ...
OnlineDriveFunctionInterface onlineDfi =
onlineDriveObject.GetService<OnlineDriveFunctionInterface>();
HardwareProjection hardwareProjection = onlineDfi.HardwareProjection;

// onlineDfi can be null in case of the actual onlineDriveObject
// does not support it or if the device is offline.
// hardwareProjection can be null, if the actual onlineDriveObject
// does not support it.
// For example: On G120 drives, you have to use the CU as onlineDriveObject. On
// S120 drives, you have to use the MotorModul as onlineDriveObject.
```

7.28.4.13 Creating a component for a drive component (S120 only)

You have the option of creating a component below a drive component for the S120.

Also enter a type designation to distinguish between the various encoders. The possible type designations and restrictions for encoders are described in the table below.

The following example shows how to create a component below a drive component.

Creating a motor and an encoder below a Motor Module

```
DeviceItem subModul = sdrDevice.PlugNew(@"OrderNumber:6SL3xxx-xxxxx-xxxx",
"MotorModul", 65535);

//Plug a motor to the motor modul
subModul.Container.PlugNew(@"OrderNumber:1PH2092-4WG4x-xxxx",
"Motor_1", 65535);

//Plug an encoder to the motor modul
subModul.Container.PlugNew(@"OrderNumber:XExxxxx-xxxxx-xxxx//DRIVE-
CLIQ.202", "Encoder_1", 65535);
```

Type designations for encoders and restrictions

The following restrictions apply when inserting encoders via Openness:

- Only an unspecific Sensor Module can be created for some encoders when inserting via Openness. In this case, you must configure the specific type of the Sensor Module in the TIA Portal.
- Maximum two encoders can be inserted for one Motor Module.

The following table lists the available type designations for encoders.

DRIVE-CLIQ	Resolver	sin/cos	SSI	sin/cos+SSI	HTL/TTL	HTL/TTL +SSI	EnDat 2.1
DRIVE-CLIQ.202	Resolver.0	SIN_COS.0	SSI.0	SIN_COS_+SSI.0	HTL_TTL.0	HTL_TTL +SSI.0	En-Dat_2.1.2051
DRIVE-CLIQ.204	Resolv-er.1001	SIN_COS.2001	SSI.3081	SIN_COS_+SSI.2081	HTL_TTL.3001	HTL_TTL +SSI.3088	En-Dat_2.1.2052
DRIVE-CLIQ.212	Resolv-er.1002	SIN_COS.2002	SSI.3082	SIN_COS_+SSI.2082	HTL_TTL.3002	HTL_TTL +SSI.3090	En-Dat_2.1.2053
DRIVE-CLIQ.214	Resolv-er.1003	SIN_COS.2003	SSI.9999	SIN_COS_+SSI.2083	HTL_TTL.3003	HTL_TTL +SSI.9999	En-Dat_2.1.2054
DRIVE-CLIQ.242	Resolv-er.1004	SIN_COS.2004	-	SIN_COS_+SSI.2084	HTL_TTL.3005	-	En-Dat_2.1.2055
DRIVE-CLIQ.244	Resolv-er.9999	SIN_COS.2005	-	SIN_COS_+SSI.9999	HTL_TTL.3006	-	En-Dat_2.1.2151
DRIVE-CLIQ.9999	-	SIN_COS.2006	-	-	HTL_TTL.3007	-	En-Dat_2.1.9999
DRIVE-CLIQ.10100	-	SIN_COS.2007	-	-	HTL_TTL.3008	-	En-Dat_2.1.10100
-	-	SIN_COS.2008	-	-	HTL_TTL.3009	-	-

DRIVE-CLiQ	Resolver	sin/cos	SSI	sin/cos+SSI	HTL/TTL	HTL/TTL+SSI	EnDat 2.1
-	-	SIN_COS.20 10	-	-	HTL_TTL.30 11	-	-
-	-	SIN_COS.20 12	-	-	HTL_TTL.30 20	-	-
-	-	SIN_COS.20 13	-	-	HTL_TTL.31 09	-	-
-	-	SIN_COS.21 10	-	-	HTL_TTL.99 99	-	-
-	-	SIN_COS.21 11	-	-	-	-	-
-	-	SIN_COS.21 12	-	-	-	-	-
-	-	SIN_COS.99 99	-	-	-	-	-

7.28.4.14 Defining the motor type and motor configuration

The following examples show how you can set a motor type for G120 drives via the device configuration:

Setting the motor type offline via the hardware configuration

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;

//Offline (Only on G120 drives)
DriveFunctionInterface dfi = ...
HardwareProjection hardwareProjection = dfi.HardwareProjection;

// hardwareProjection can be null in case of the actual driveObject
// does not support activation.
// Setting the required MotorType and DriveDataSetNumber on the drive.
// It is only supported on G120 drives.

//First parameter is the MotorType, Second parameter is the DriveDataSetNumber
hardwareProjection.SetMotorType(MotorType.InductionMotor, 0);

// There can be a problem, if the selected MotorType is not available
// on the drive. In that case, it has to give back a feedback.
```

Setting the motor type online via the hardware configuration

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;
```

Setting the motor type online via the hardware configuration

```
//Online (Only on G120 drives)
OnlineDriveFunctionInterface onlineDfi = ...
HardwareProjection hardwareProjection = onlineDfi.HardwareProjection;

// hardwareProjection can be null in case of the actual
// onlineDriveObject does not support activation.
// Setting the required MotorType and DriveDataSetNumber on
// the drive. It is only supported on G120 drives.

// First parameter is the MotorType, Second parameter is the DriveDataSetNumber
hardwareProjection.SetMotorType(MotorType.InductionMotor, 0);

// There can be a problem, if the selected MotorType is not available on
// the drive. In that case, it has to give back a feedback.
```

The following examples show how you can read out the motor type for G120 drives from the drive:

Determining the motor configuration offline

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;

// Offline
// WARNING: You have to set the MotorType on G drives before
// you would like to get the current configuration, otherwise
// you will get an exception/feedback, which informs you that
// there is no motor set.
// On S120 drives, you don't have to do that, but you have to
// plug a motor to motor modul.( later on)

HardwareProjection hardwareProjection = dfi.HardwareProjection;
// Get the current motor configuration
// It needs a datasetnumber, which is currently only supported on G120 drives.
MotorConfiguration motorConfiguration = hardwareProjection.GetCurrentMotorConfiguration(0);
```

Determining the motor configuration online

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;

// Online
// WARNING: You have to set the MotorType on G drives before
// you would like to get the current configuration, otherwise
// you will get an exception/feedback, which informs you that
// there is no motor set.
// On S120 drives, it is not possible??

HardwareProjection hardwareProjection = onlineDfi.HardwareProjection;
// Get the current motor configuration
// It needs a datasetnumber, which is currently only supported on G120 drives.
MotorConfiguration motorConfiguration = hardwareProjection.GetCurrentMotorConfiguration(0);
```

The following example shows how you can create a motor configuration for G120 drives:

Configuring a motor configuration

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;

DeviceItem cuDeviceItem = m_Device.DeviceItems[1];
DriveObject cuDriveObject =
cuDeviceItem.GetService<DriveObjectContainer>().DriveObjects[0];
DriveFunctionInterface cuDriveFunctionInterface =
cuDriveObject.GetService<DriveFunctionInterface>();

HardwareProjection hardwareProjection = cuDriveFunctionInterface.HardwareProjection;
hardwareProjection.SetMotorType(MotorType.InductionMotor, 0);

MotorConfiguration config = hardwareProjection.GetCurrentMotorConfiguration(0);

config.RequiredConfigurationEntries.ToList().ForEach(ce =>
{
    switch (ce.Number)
    {
        case 305:
            ce.Value = 20;
            break;
        case 307:
            ce.Value = 30;
            break;
        case 311:
            ce.Value = 200;
            break;
        case 304:
            ce.Value = 450;
            break;
        case 310:
            ce.Value = 50;
            break;
        case 335:
            ce.Value = 1;
            break;
        default:
            break;
    }
});
bool result =
cuDriveFunctionInterface.HardwareProjection.ProjectMotorConfiguration(config, 0);
```

7.28.4.15 Reading and writing parameters

The following example shows how to read and write values of drive parameters. You require a drive object for access.

Access to parameters

```
using Siemens.Engineering.MC.Drives;
```

Access to parameters

```

//Access a parameter via its name
DriveParameter parameter = driveObject.Parameters.Find("p5391[0]");

//Example of reading parameter attributes
if (parameter != null)
{
    Console.WriteLine("The Name of the parameter is : " + parameter.Name);
    Console.WriteLine("The value of the parameter is : " +
parameter.Value.ToString());
    Console.WriteLine("The minValue of the parameter is : " +
parameter.MinValue);
    Console.WriteLine("The MaxValue of the parameter is : " +
parameter.MaxValue);
    Console.WriteLine("The Unit of the parameter is : " + parameter.Unit);

    //Example for write:
    parameter.Value = 60;
}

// Access a parameter via Number and ArrayIndex// Note that
// - arrayless parameters (e.g. p96 on G120) are indexed by -1
// - parameters that consist of only a bit array do not count as
// array parameters, they are indexed by -1 as well and the
// further bit values can be accessed by the 'Bits' property
// of the given parameter (e.g. r2139 on G120). For further
// information about accessing bit parameters see the next
// section of this code snippet

DriveParameter r947_6 = driveObject.Parameters.Find(947, 6); // returns
r947[6]if (r947_6 != null)
{
    Console.WriteLine("The Name of the parameter is : " + r947_6.Name);
    Console.WriteLine("The value of the parameter is : " +
r947_6.Value.ToString());
}

//Accessing bit values
DriveParameter p2720 = driveObject.Parameters.Find(2720, 0);if (p2720 !=
null)
{
    // Note that in general, pXXX.Bits[YYY] is not necessarily equivalent to
    pXXX.YYY.
    // pXXX.Bits is an array of available bit values in ascending order by
    their names
    // and not an array of bit values indexed by their names.
    // For example: let the available bit values be [pXXX.0, pXXX.2, pXXX.3],
then
    // pXXX.Bits[2] == pXXX.3, not pXXX.2

    DriveParameter p2720Bit1 = p2720.Bits[1]; // returns p2720[0].1
    if (p2720Bit1 != null)
    {
        Console.WriteLine("The name of the parameter is : " + p2720Bit1.Name);
        Console.WriteLine("The value of the second bit of the parameter is : " +
+ p2720Bit1.Value.ToString());
    }
}

```

```
Access to parameters
}

//Get the enum values of a parameter
DriveParameter r47 = driveObject.Parameters.Find("r47");
foreach (var enumItem in r47.EnumValueList)
{
    Console.WriteLine("Enum value: " + enumItem.Key.ToString() + " = " +
enumItem.Value);
}
```

7.28.4.16 Reading and writing parameters online

The following example shows how to obtain a list with the available online parameters. You require an online drive object for access.

The read and write access to individual online parameters from the parameter list is identical to that in the Reading and writing parameters (Page 720) example.

Access to online parameters

```
using Siemens.Engineering.MC.Drives;
DeviceItem item = ... //device item of the CU (e.g. :
project.Devices[0].Items[0].Items[0])
OnlineDriveObject onlineDriveObject =
item.GetService<OnlineDriveObjectContainer>().OnlineDriveObjects[0];
if (onlineDriveObject != null)
{
    var parameters = onlineDriveObject.Parameters;
}
```

7.28.4.17 Saving the parameterization

The following example shows how you can determine the parameterization from S120, S210 or G120 drives.

Determining the parameterization online from the drives

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;
```

Determining the parameterization online from the drives

```
OnlineDriveObject onlineDriveObject =
item.GetService<OnlineDriveObjectContainer>().OnlineDriveObjects[0];
OnlineDriveFunctionInterface onlineDfi =
onlineDriveObject.GetService<OnlineDriveFunctionInterface>();
DriveDomainFunctions driveDomainFunctions = onlineDfi.DriveDomainFunctions;

// onlineDfi can be null in case of the actual onlineDriveObject
// does not support it or if the device is offline.

// driveDomainFunctions can be null, if the actual onlineDriveObject
// does not support it.

// For example: On G120 and S210 drives, you have to use the CU to get the
onlineDriveObject.
// Please, pay attention at S120 drives. Here you can use any module (CU, MotorModul,
lineModul) as onlineDriveObject but preferable the CU, because it will run on CU!
```

The following example shows how you can copy the parameterization for S120, S210 or G120 drives from RAM to ROM - and therefore retentively save the parameter assignment.

Backing up from RAM to ROM

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;
//In case of G120 device.
DeviceItem cuDeviceItem = m_Device.DeviceItems[1];
OnlineDriveObject cuDriveObject =
cuDeviceItem.GetService<OnlineDriveObjectContainer>().OnlineDriveObjects[0];

//In case of S120 and S210 device you should generally get the CU driveobject.
DeviceItem cuDeviceItem = m_Device.DeviceItems[0];OnlineDriveObject cuDriveObject =
cuDeviceItem.GetService<OnlineDriveObjectContainer>().OnlineDriveObjects[0];

// To use the function, you have to use the OnlineDriveFunctionInterface.
OnlineDriveFunctionInterface cuDriveFunctionInterface =
cuDriveObject.GetService<OnlineDriveFunctionInterface>();
DriveDomainFunctions driveDomainFunctions = cuDriveFunctionInterface.DriveDomainFunctions;

// This function will perform a RAM to ROM copy on all device.
// It is important that the CU should be in online state before you call this function
anyway the program will give an exception.
// This call may take a few moments before be finished.
bool result = DriveDomainFunctions.PerformRAMtoROMCopyAllDriveObject();
```

7.28.4.18 Using Safety Integrated telegrams

The following example shows how you can use Safety Integrated telegrams (e.g. 30) in Openness.

Using Safety Integrated telegrams

```
using Siemens.Engineering.MC.Drives;
```

Using Safety Integrated telegrams

```
TelegramComposition telegrams = drvObj.Telegrams;

//Add safety telegram
const int tgrmNumber = 30;
drvObj.Telegrams.InsertSafetyTelegram(tgrmNumber);

//Find safety telegram
Telegram safetyTgrm = drvObj.Telegrams.Find(TelegramType.SafetyTelegram);

// Get and set safety telegram attributes
uint watchDogTime = (uint)safetyTgrm.GetAttribute("Failsafe_FMonitoringtime");

safetyTgrm.SetAttribute("Failsafe_FMonitoringtime", 300);

const int newSafetyTelegramNumber= 900;
if (safetyTgrm.CanChangeTelegram(newSafetyTelegramNumber))
{
    safetyTgrm.TelegramNumber = newSafetyTelegramNumber;
}

//Remove Safety telegram
drvObj.Telegrams.EraseTelegram(TelegramType.SafetyTelegram);
```

7.28.4.19 Inserting and extending telegrams

The following example shows how to insert an extension and change the size of a standard telegram. You require a drive object for access.

Inserting an extension and changing the size of a standard telegram
using Siemens.Engineering.MC.Drives;

Inserting an extension and changing the size of a standard telegram

```

TelegramComposition telegrams = drvObj.Telegrams;
Telegram telegram = telegrams.Find(TelegramType.MainTelegram);

Console.WriteLine("The Cu has the telegram: " + telegram.TelegramNumber);
Console.WriteLine("The Setpoint channel-specific size of the telegram is: "
+ telegram.GetOutputSize());

foreach (var address in telegram.Addresses)
{
    if (address.IoType == AddressIoType.Output)
    {
        Console.WriteLine("The Setpoint channel-specific starting address of
the connected PLC is: " + address.StartAddress);
    }
    else if (address.IoType == AddressIoType.Input)
    {
        Console.WriteLine("The Actual value channel-specific starting address
of the connected PLC is: " + address.StartAddress);
    }
}

// Add an additional Telegram
if (drvObj.Telegrams.CanInsertAdditionalTelegram(2, 4))
{
    drvObj.Telegrams.InsertAdditionalTelegram(2, 4);
}

// Add a 3 word extension to the main telegram
Telegram mainTelegram == drvObj.Telegrams.Find(TelegramType.MainTelegram);
Int32 newSize = mainTelegram.GetSize(AddressIoType.Input) + 3;
if (mainTelegram.CanChangeSize(AddressIoType.Input, newSize, true))
{
    mainTelegram.ChangeSize(AddressIoType.Input, newSize, true)
}

```

7.28.4.20 Using torque telegrams

The following example shows how you can use torque telegrams (e.g. 750) in Openness.

Using torque telegrams

```

using Siemens.Engineering.MC.Drives;
const int torqueTelegramNumber = 750;
TelegramComposition telegrams = drvObj.Telegrams;

// Add a new torque telegram
if (drvObj.Telegrams.CanInsertTorqueTelegram(torqueTelegramNumber))
{
    drvObj.Telegrams.InsertTorqueTelegram(torqueTelegramNumber);
}

// Find torque telegram
Telegram torqueTelegram = drvObj.Telegrams.Find(TelegramType.TorqueTelegram);

```

7.28.4.21 Restoring factory settings

The following example shows how you can restore the factory settings for S120, S210 or G120 drives.

Restoring factory settings

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;
OnlineDriveObject onlineDriveObject = ...
OnlineDriveFunctionInterface onlineDfi =
onlineDriveObject.GetService<OnlineDriveFunctionInterface>();
DriveDomainFunctions driveDomainFunctions = onlineDfi.DriveDomainFunctions;

// onlineDfi can be null in case of the actual onlineDriveObject
// does not support it or if the device is offline.
// saveParametrization can be null, if the actual onlineDriveObject
// does not support it.
// For example: On G120 and S210 drives, you have to use the CU to get the
onlineDriveObject.
// S120 drives, you can use any modul (CU, MotorModul, lineModul) as onlineDriveObject but
preferable the CU.
```

7.29 Functions for DCC

7.29.1 Introduction

Using TIA Portal-Openness, you automate the engineering and control the TIA Portal using a program that you created yourself.

In this help document, you can find a lot of information and code examples for this program that you generate yourself. You can also generate and use your own programs for the TIA Portal "DCC" application.

Before you generate your own program for DCC from the sample codes listed in the following, please note the general information on Openness, which you can find in this help under the following keywords:

- Preconditions for using TIA Portal Openness
- Installing TIA Portal Openness
- Accessing the TIA Portal
- TIA Portal Openness object model
- Programming steps

7.29.2 DCC Openness

Introduction

With TIA Portal Openness V16, you program applications that are automated in TIA Portal engineering.

Openness functions in conjunction with SINAMICS DCC

The following functions have been implemented in Version V16:

- Import charts (Page 733)
- Export one chart (Page 733) or Export all charts ("Charts" folder) (Page 733)
- Import DCB Extension libraries into the project library (Page 736)
- Find charts based on their names (Page 734)
- Delete a chart from the "Charts" folder (Page 735)
- Optimize the block run sequence in a chart (Page 735)

Note

Subcharts

Charts that contain subcharts can be imported using Openness. However, Openness cannot be used to access subcharts.

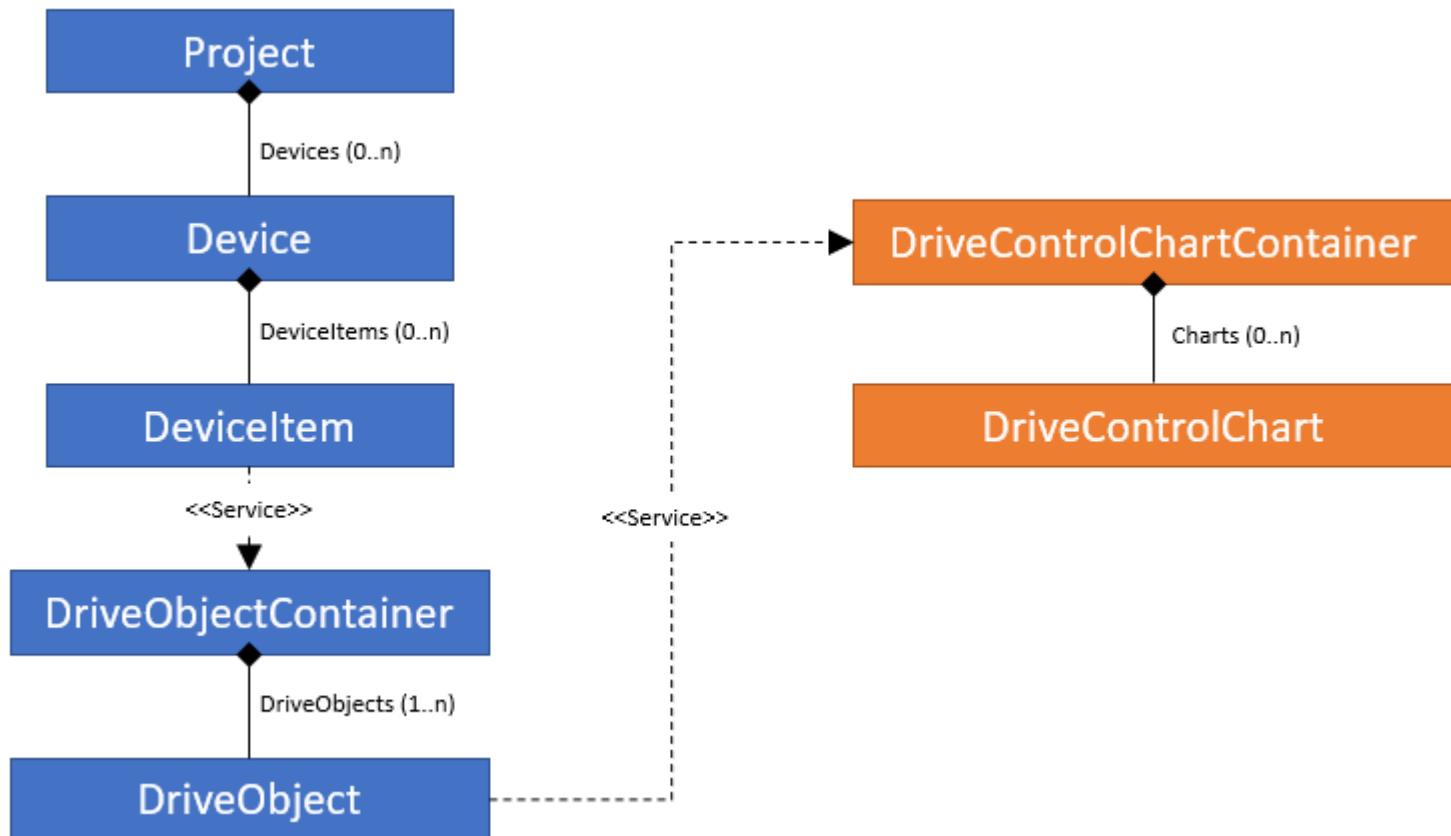
Additional information

See TIA information system "Openness: Automate project creation".

7.29.3 DCC Openness object model

Overview

The following diagram describes the DCC Openness object model:



7.29.4 References

7.29.4.1 DriveControlChartContainer

DriveParameter

DriveControlChartContainer is a service that represents the DCC chart container under a DriveObject, so that it can be retrieved from the appropriate DriveObject. The service is not provided if the DriveObject is not from a supported device.

The following table describes the **Navigators** of the DriveControlChartContainer:

Name	Data type	Writable	Description
Charts	DriveControlChartComposition	Read only	Reads out the chart layout (composition) in the chart container.

7.29.4.2 DriveControlChartComposition

DriveControlChartComposition features

DriveControlChartComposition lists the available charts.

The following table describes the **Methods** of DriveControlChartComposition:

Name	Type of return value	Parameter	Description	Exceptions
Import	DccImportResultData (Page 730)	string path, DccImportOptions (Page 730) importOptions	Method description: Imports charts from a DCC export file. Parameter description: path: Complete path of the file to be imported. importOptions: Options used when importing, see DccImportOptions (Page 730).	DccImportException (Page 739)
Export	-	string path	Method description: Exports all charts to a DCC export file. Parameter description: path: Complete path of the file to be exported.	DccExportException (Page 739)
GetChartSequence	DriveControlChart (Page 730)	-	Reads out all charts in the container in the order in which they are run.	-
Find	DriveControlChart (Page 730)	string name	Identifies a chart based on its name in the chart container.	-

See also

- DriveControlChartComposition exceptions (Page 739)
- DccImportResultData (Page 730)
- DriveControlChart (Page 730)
- DccImportOptions (Page 730)

7.29.4.3 DcclImportOptions

DcclImportOptions

These settings are possible when importing charts:

- DcclImportOptions.None:
The import has been successfully completed if there is no name conflict. Otherwise exception DcclImportChartWithSameNameAlreadyAvailableException is thrown.
- DcclImportOptions.RenameOnConflict:
If there is a chart name conflict, the CFC automatically assigns a name to the newly imported chart. For example, if a chart with the name CFC_1 already exists on import, then the newly imported chart is assigned name CFC_2.

See also

[DriveControlChartComposition \(Page 729\)](#)

7.29.4.4 DcclImportResultData

DriveParameter

DcclImportResultData contains information about the result when importing charts.

The following table describes the **attributes** of DcclImportResultData:

Attribute name	Data type	Writable	Description
RemappedParameterNumbers	IDictionary<UInt16, UInt16>	Read only	When importing, it is possible that a DCC parameter to be imported is already available. In this case, the imported parameter is assigned a new number. This dictionary then contains all of the newly assigned parameters. The key value is the previous parameter number as it was exported and the value is the newly created parameter.

See also

[DriveControlChartComposition \(Page 729\)](#)

7.29.4.5 DriveControlChart

DriveControlChart Features

DriveControlChart corresponds to a chart within the chart container (Openness does not support subcharts).

The following table describes the **attributes** of the DriveControlChart:

Attribute name	Data type	Writable	Description
Name	string	Read only	Name of the chart.

The following table describes the **methods** of DriveControlChart:

Name	Type of return value	Parameter	Description	Exceptions
Export	-	string path	Method description: Exports only this chart. Parameter description: path: Complete path of the file to be exported.	DccExportException (Page 738)
Delete	-	-	Method description: Deletes the chart.	-
OptimizeRunSequence	-	-	Description method: Optimizes the run sequence of the chart. CFC provides the optimization mechanism. Note: The run sequence can only be optimized if a DCC license is available.	DccLicenseUnavailableException (Page 738)

See also

[DriveControlChart Exceptions \(Page 738\)](#)

[DriveControlChartComposition \(Page 729\)](#)

[Deleting a chart \(Page 735\)](#)

7.29.4.6 Importing DCB extension library

ImportDCBextensionlibrary

The following table describes the **Methods** of ImportDCBextensionlibrary:

Name	Return value	Parameter	Description
ImportDcbLibrary	-	string path	Method arguments: path: The complete path of the DCB Extension library zip file. Method description: Imports a DCB Extension library into the project library.

7.29.5 Code examples

7.29.5.1 General

The following code examples describe the basic procedure for various applications. The code is not necessarily complete or compilable.

7.29.5.2 Accessing a DriveControlChartContainer via DriveObject

The following example shows how you can access a DriveControlChartContainer via a drive object.

Accessing a DriveControlChartContainer

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DCC;
Project project = portal.Projects.Open("..."); //Destination folder to
open the project
DeviceItem item = project.Devices[0].Items[0];
DriveObject
driveObject = item.GetService<DriveObjectContainer>().DriveObjects[0];
DriveControlChartContainer
chartContainer = driveObject.GetService<DriveControlChartContainer>();
//chartContainer can be null in case the DriveObject does not support DCC.
```

7.29.5.3 Retrieving charts

The following example shows how you can retrieve charts.

Retrieving charts

```
DriveControlChartComposition charts = chartContainer.Charts;
```

7.29.5.4 Accessing charts

The following example shows how you can access charts.

Accessing charts

```
DriveControlChartComposition charts = ...
DriveControlChart firstChart = charts[0];

// or looping...
foreach(DriveControlChart chart in charts)
{
    ...
}
```

7.29.5.5 Importing charts

The following example shows how you can import charts.

Importing charts

```
try
{
    DriveControlChartComposition charts = ...
    DccImportResultData result = charts.Import(@"c:\Charts.dcc",
DccImportOptions.None);
}
catch (DccImportException exc)
{
}
```

See also

DCC Openness (Page 727)

7.29.5.6 Exporting charts

The following example shows how you can export charts.

Exporting charts

```
try
{
    DriveControlChart chart;
    ...
    chart.Export(@"c:\CFC_1.dcc");
}
catch (DccExportException exc)
{
}
```

See also

DCC Openness (Page 727)

7.29.5.7 Export all charts

The following example shows how you can export all charts.

Exporting all charts

Exporting all charts

```
try
{
    charts.Export(@"c:\Charts.dcc");
}
catch (DccExportException exc)
{
}
```

See also

[DCC Openness \(Page 727\)](#)

7.29.5.8 Retrieving charts in the order in which they are executed

The following example shows how you can retrieve charts in the order in which they are executed.

Retrieving charts

```
DriveControlChartComposition charts = ...
IList<DriveControlChart> chartSequence = charts.GetChartSequence();
```

7.29.5.9 Finding charts based on their names

The following example shows how you can find a chart based on its name.

Finding charts

```
DriveControlChartComposition charts = ...
DriveControlChart chart = charts.Find("CFC_1");
```

See also

[DCC Openness \(Page 727\)](#)

7.29.5.10 Displaying an import report

The following example shows how you can display a report associated with the import.

Displaying an import report

Displaying an import report

```
DccImportResultData importResultData = charts.Import(@"d:\Charts.dcc",
DccImportOptions.None);
IDictionary<ushort, ushort> remappedParameters = importResultData.RemappedParameterNumbers;
foreach (KeyValuePair<ushort, ushort> remappedParameter in remappedParameters)
{
    System.Console.WriteLine($"ParameterNumber <{remappedParameter.Key}> has
been changed to <{remappedParameter.Value}> during import.");
}
```

7.29.5.11 Deleting a chart

The following example shows how you can delete charts.

Deleting charts

```
try
{
    DriveControlChart chart;
    ...
    chart.Delete();
}
```

See also

[DCC Openness \(Page 727\)](#)

[DriveControlChart \(Page 730\)](#)

7.29.5.12 Optimizing the execution sequence

The following example shows how you can optimize the sequence in which charts are run.

Optimizing the run sequence of charts

```
try
{
    DriveControlChart chart;
    ...
    chart.OptimizeRunSequence();
}
catch (DccLicenseUnavailableException exc)
{}
```

See also

[DCC Openness \(Page 727\)](#)

7.29.5.13 Importing a DCB Extension library

The following example shows how you can import a DCB Extension library.

Importing a DCB Extension library

```
try
{
    Project myProject;
    ...

    myProject.ProjectLibrary.ImportDcbLibrary(@"c:\GMCV5_1_sinamics5_1_(5.1.15
).zip");
}
catch (DccImportException exc)
{
}
```

See also

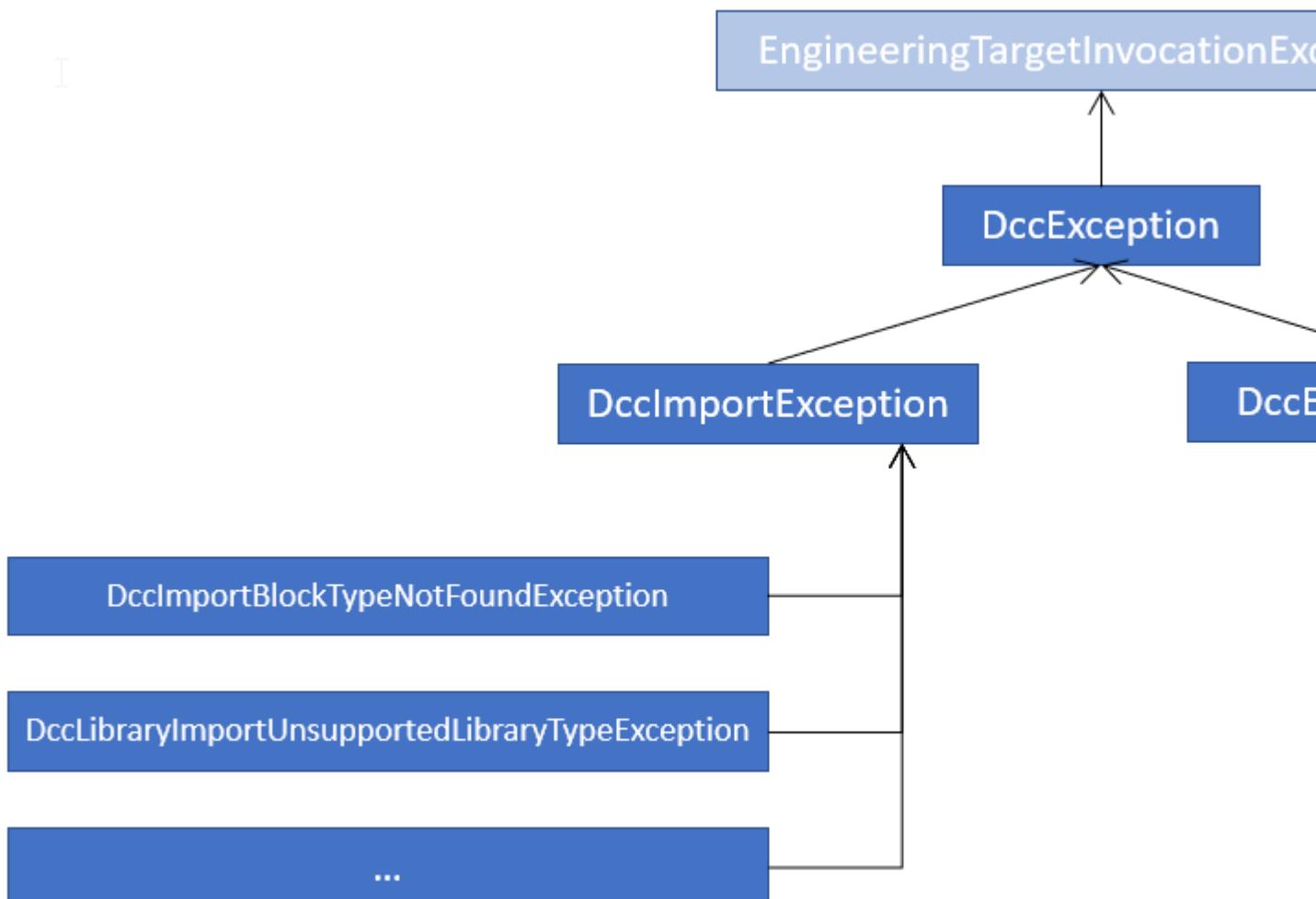
[DCC Openness \(Page 727\)](#)

7.29.6 DCC Openness exceptions

7.29.6.1 Exception handling

Exception handling

For applications that could fail because of user error, DCC Openness throws a DCC specific exception derived from `EngineeringTargetInvocationException`. These exceptions are hierarchical, which means that all specific exceptions are derived from the higher-level exception `DccException`:



If client does not want to evaluate or distinguish between different error causes, then the simplest way is to catch the general `DccException`:

```

try
{
    Project myProject;
    ...

    myProject.ProjectLibrary.ImportDcbLibrary(@"c:\GMCV5_1_sinamics5_1_(5.1.15
).zip");
    ...
    DriveControlChartContainer chartContainer = driveObject.GetService<DriveC
ontrolChartContainer>();
    chartContainer.Charts.Import(@"d:\Charts.dcc", DccImportOptions.None);
}
catch (DccException exc)
{
}

```

If you wish to respond based on the type of error, then catch all relevant exceptions:

```

try
{
    DriveControlChartContainer
chartContainer = driveObject.GetService<DriveControlChartContainer>();
    chartContainer.Charts.Import(@"d:\Charts.dcc", DccImportOptions.None);
}
catch (DccImportLibraryIsMissingException missingLibExc)
{
}
catch (DccImportChartWithSameNameAlreadyAvailableException sameNameExc)
{
}
catch (DccImportException exc)
{
}
catch (DccException exc)
{
}

```

7.29.6.2 DriveControlChart Exceptions

Exceptions

Exceptions that can be thrown when exporting:

- **DccExportException:**
Is thrown for all export error types.

Exceptions that can be thrown when executing the function to optimize the sequence in which charts are run.

- **DccLicenseUnavailableException:**
Is thrown if a DCC license is not available. In this case, the function to optimize the run sequence is not available.

See also

[DriveControlChart \(Page 730\)](#)

7.29.6.3 DriveControlChartComposition exceptions

Exceptions

Exceptions that can be thrown when importing:

- **DcImportException:**
General import exception, which DCC throws in the case of an import error.
- **DcImportBlockCreationException:**
Is thrown if a block was not able to be created when being imported.
- **DcImportBlockTypeNotFoundException:**
Is thrown if a referenced DCB block type was not able to be found in a standard DCBLib or in an extension library.
- **DcImportChartCreationException:**
- Is thrown if a chart was not able to be created when being imported.
- **DcImportChartWithSameNameAlreadyAvailableException:**
Is thrown if a chart with the same name already exists in the chart container and `DcImportOptions.None` is used.
- **DcImportLibraryIsMissingException:**
Is thrown if a referenced DCB Extension library does not exist in the project.
- **DcImportFileAlreadyInUseException:**
Is thrown if the file being imported is already being used in another process.
- **DcImportDcbTypeDifferentVersionAlreadyUsedException:**
Is thrown if a DCB type in a different version of the same DCB Extension library is already being used in the device.

See also

[DriveControlChartComposition \(Page 729\)](#)

7.29.6.4 ImportDcbLibrary Exceptions

Exceptions

Exceptions thrown by ImportDcbLibrary:

- **DcImportException:**
Is thrown for all import error types.
- **DccLibraryImportAlreadyAvailableException:**
Is thrown if the same version of the DCB Extension library already exists in the project.
- **DccLibraryImportCorruptedStudioLibraryException:**
Is thrown if the DCB Extension library zip file is corrupted and cannot be imported.
- **DccLibraryImportIntegrityBrokenException:**
Is thrown if the zip file of the DCB Extension library is corrupted.
- **DccLibraryImportOverallPinLimitExceededException:**
Is thrown if the DCB Extension library contains a DCB type that exceeds the maximum number of pins permitted.
- **DccLibraryImportStandardLibraryAlreadyAvailableException:**
Is thrown if the DCB Extension library selected for import is the standard DCBLib, and already exists.
- **DccLibraryImportUnsupportedLibraryTypeException:**
Is thrown if the selected DCB Extension library is not supported by DCC and SINAMICS drives.

7.30 Exceptions

7.30.1 Handling exceptions

Exceptions when accessing the TIA Portal via TIA Portal Openness APIs

During the execution of an TIA Portal Openness application via the TIA Portal Openness API, all errors which occur are reported as exceptions. These exceptions contain information that will help you to correct the errors which have occurred.

A distinction is made between two types of exceptions:

- Recoverable (`Siemens.Engineering.EngineeringException`)

You can continue to access the TIA Portal without interruption with this exception.

Alternatively, you can cancel the connection to the TIA Portal.

The EngineeringExceptions include the following types:

- Security-related exceptions (`EngineeringSecurityException`), for example, in case of missing access rights.
- Exceptions when accessing objects (`EngineeringObjectDisposedException`), for example, when accessing objects which no longer exist.
- Exceptions when accessing attributes (`EngineeringNotSupportedException`), for example, when accessing attributes which do not exist.
- General exceptions when calling (`EngineeringTargetInvocationException`), for example, error despite valid call of TIA Portal Openness API.
- Exceptions when calling (`EngineeringRuntimeException`), for example, invalid cast exception.
- Exceptions when there are not enough resources in associated TIA Portal instance (`EngineeringOutOfMemoryException`)
- Exceptions when calling is terminated (`EngineeringUserAbortException`), for example, during an import action canceled by user.
- Exceptions thrown during the API call invoked from a client provided delegate (`EngineeringDelegateInvocationException`). This exception is derived from `EngineeringTargetInvocationException` exception.

The EngineeringExceptions have the following attributes:

- `ExceptionMessageData messageData`: Contains the reason for which the exception was thrown.

- `ExceptionMessageData detailMessageData`: Contains additional information about the reason. The result is returned as a `<IList>`.

- `String message`: Returns the result from `MessageData` and `DetailMessageData`.
`ExceptionMessageData` returns the following information:

- `String Text`: Contains the reason for which the exception was thrown.

- NonRecoverable (`Siemens.Engineering.NonrecoverableException`)

This exception closes the TIA Portal and the connection to the TIA Portal is disconnected. You need to restart the TIA Portal using the TIA Portal Openness application.

Program code

The following example shows the options you have for responding to exceptions:

```
try
{
    ...
}

catch(EngineeringSecurityException engineeringSecurityException)
{
    Console.WriteLine(engineeringSecurityException);
}

catch(EngineeringObjectDisposedException engineeringObjectDisposedException)
{
    Console.WriteLine(engineeringObjectDisposedException.Message);
}

catch(EngineeringNotSupportedException engineeringNotSupportedException)
{
    Console.WriteLine(engineeringNotSupportedException.MessageData.Text);
    Console.WriteLine();
    foreach(ExceptionMessageData detailMessageData in
engineeringNotSupportedException.DetailMessageData)
    {
        Console.WriteLine(detailMessageData.Text);
    }
}

catch (EngineeringTargetException)
{
    throw;
}

catch (EngineeringException)
{
    //Do not catch general exceptions
    throw;
}

catch(NonRecoverableException nonRecoverableException)
{
    Console.WriteLine(nonRecoverableException.Message);
}
```

7.30.2 Custom exception

Introduction

CustomException is a mechanism that allows Openness API Developer to have a rich set of exceptions to handle error scenarios to implement a long term stable error handling. Through CustomException support, Openness API Developer can create their own Custom Exceptions in the EOM via EOM Designer, and Openness API users catch those exceptions in the Openness API code in addition to the predefined ones. Custom exception defined in EOM should be attached to an Openness action which can throw this custom exception so that you can catch custom exception. Otherwise, you will get EngineeringTargetInvocationException.

CustomException in EOM Designer

The EOM modeler creates CustomException using EOM designer. The exception can be created under any namespace.

CustomException in Openness application

You can use the TIA Portal Openness application to directly catch CustomException using the Openness code:

```
static void Main(string[] args)
{
try
{
SomeEOMType someEOMType = // Get some EOM type.
someEOMType.ActionThatThrowsException();
}
catch (AnEomCustomException ex)
{
//if AnEomCustomException is attached to someEOMType.ActionThatThrowsException() as
throwable exception.
}
catch (EngineeringTargetInvocationException ex)
{
}
}
```

New EOM type

CustomException is a new type in EOM meta, but it is a class in Siemens.Engineering.dll. CustomException in EOM looks like below:

```
<CustomException name="MaxCharactersExceededException" publicationLevel="Published"
createPublicationLevel="Published" accessModifier="Public">
<Product>Automation-Main</Product>
<Extends>
<Class name="Siemens.Engineering.EngineeringTargetInvocationException" />
</Extends>
<Mapping
ServerException="Siemens.Automation.CustomIdentity.BusinessLogic.Exceptions.MaxCharactersE
xceededException"> </CustomException>
```

Mapping

EOM modeler should map ServerException of a CustomException to a server exception in client component in-order to throw the CustomException in Openness code for Openness API users when the client component throws the server exception. The server exception is a full name of the exception in server (client component). The server exception defined in the client should be derived from Siemens.Automation.CommonServices.UserExceptionBase.

Publication level

The Publication level for CustomException cannot be Elevated or System, and Publication level for Openness system Exception should be Published.

Extends

The CustomException which you creates should derive from either EngineeringTargetInvocationException or derivation of EngineeringTargetInvocationException. The EOM designer provides the option to extend the exception.

Rules

For EOM modelers, the EOM designer shows errors and generator fails to generate engineering assembly when the following scenarios are violated.

- Errors:
 - The custom Exception name should end with "Exception". Eg: MaxCharactersExceededException
 - The access modifier for a custom exception should be public.
 - It should be derived from either EngineeringTargetInvocationException or from any other derivation of EngineeringTargetInvocationException
 - ServerException mapping is required for an exception to throw the exception in client side. Otherwise the exception will be available in EOM meta but no one can throw it.
 - ServerException provided should be unique.
 - Publication level for CustomException cannot be Elevated or System.
- Info
 - If the Throwable exception is attached to a Create action then it will be used only for Create action documentation

These rules are applicable only for custom exceptions created by Openness users. Eg: SampleException.

The above rules are not applicable for below mentioned Openness system exceptions

- EngineeringException
- EngineeringSecurityException
- EngineeringObjectDisposedException
- EngineeringNotSupportedException
- EngineeringTargetInvocationException
- EngineeringRuntimeException
- EngineeringOutOfMemoryException
- EngineeringUserAbortException
- EngineeringDelegateInvocationException
- NonRecoverableException

Throwable Exception of an Action

Openness API developer can attach a CustomException to an action through Openness designer. Openness System exceptions like EngineeringTargetInvocationException, EngineeringObjectDisposedException etc are restricted to attach to an action. You are able to catch CustomException only if the server exception throws from TiaPortal server component should have an EOM CustomException mapping and the CustomException should be attached to the invoked action.

If a custom exception is attached to a special actions like delete or create actions then the throwable exception is only used for API documentation. If the exception from the client

7.30 Exceptions

component has a mapped EOM Custom Exception in EOM side then you will be able to catch the CustomException for special actions (Create action).

Rules for throwable exception

- Only Published, Developer & Pilot levels are allowed for Custom Exception
- Elevated and System levels are not allowed for Custom Exception
- Custom Exception is allowed to attach an actions as per the levels below.

EOM action publication level	Custom exception level	Is Custom exception allowed on Action
Published	Published	Yes
Published	Developer	No
Published	Pilot	No
Developer	Developer	Yes
Developer	Published	Yes
Developer	Pilot	No
Pilot	Pilot	Yes
Pilot	Published	Yes
Pilot	Developer	No
Elevated	Pilot	No
Elevated	Published	Yes
Elevated	Developer	No
System	Published	Yes
System	Pilot	No
System	Developer	No

Note

- OPNS Designer shows an error when the publication level of EOM Action & Custom Exception is not compatible
 - Generation of Engineering assembly shall fail when there are publication level compatible errors.
-

Export/import

8.1 Overview

8.1.1 Basic principles of importing/exporting

Introduction

You can export certain configuration data and then re-import the data to the same or a different project after editing.

Note

There are no obligations or guarantees of any kind associated with using this description to manually modify and evaluate the source file. Siemens therefore accepts no liability arising from the use of all or part of this description.

Exportable and importable objects

The following configuration data can also be imported or exported by means of TIA Portal Openness APIs:

Table 8-1 Projects

Objects	Export	Import
Project graphics	X	X

Table 8-2 PLC

Objects	Export	Import
Blocks	X	X
Know-how protected blocks	X	-
Failsafe blocks	X	X
System blocks	X	-
PLC tag tables	X	X
Technology objects	X	X
PLC tags and constants	X	X
User data types	X	X

Table 8-3 HMI

Objects	Export	Import
Screens	X	X
Screen templates	X	X
Global screens	X	X
Pop-up screens	X	X
Slide-in screens	X	X
Scripts	X	X
Text lists	X	X
Graphic lists	X	X
Cycles	X	X
Connections	X	X
Tag table	X	X
Tags	X	X

Complete export or export of open references

The object types listed above are exported or imported along with all objects if these belong to the same sub-tree. This rule is also valid for referenced objects of the same sub-tree.

For referenced objects in other sub-trees, however, a complete export or import is not possible. Instead, "open references" to these objects are exported or imported.

Referenced objects of the same sub-tree are only exported if they belong to the group of exportable objects. Any dynamizations on objects are treated as objects during the import/export, and are exported and imported as well.

The export includes all object attributes that were changed during configuration. This applies regardless of whether the altered attribute will be used or not.

Example: You have configured a graphic IO field with the mode "Input/Output" and selected the setting "Visible after clicking" for the attribute "Scroll bar type". In the course of configuration, you have changed the mode to "Two states". The attribute "Scroll bar type" is not available in this mode. Because the "Scroll bar type" attribute was changed, it is included in the export, even though the attribute is not used.

Importing open references

You can also import objects with open references (see Importing configuration data (Page 753)).

If the referenced objects are contained in the target project, the open references are automatically linked to the object types again. These objects must be available at the same location and be assigned to the same name as for the export. If the referenced objects are not contained in the target project the open references can't be resolved. No additional object will be created to resolve these open references.

Export and import file format

The export and import file format is XML. Only CAx data require AML format. The different schema definitions for all formats are described in the respective section of this manual:

- XML format for the data of a HMI devices (Page 761)
- XML format for the data of a PLC devices (Page 812)
- AML format for CAx data (Page 889)

Importing and exporting fonts

Fonts defined on objects are also exported and imported.

When you import fonts that are not included in the project, the standard font is displayed at the object after the import. However, the imported font is stored in the data management.

If the attributes for a font are not assigned in an import file, the attributes are assigned default values after the import.

Importing and exporting technology objects

The following technology objects with version \geq V5.0 can be exported and imported as of TIA Portal \geq V16 for S7-1500 and S7-1500T:

- SpeedAxis
- PositioningAxis
- SynchronousAxis
- ExternalEncoder
- Cam
- OutputCam
- CamTrack
- Kinematics
- LeadingAxisProxy

The following technology objects can be exported and imported as of TIA Portal \geq V16 for S7-1200:

- PositioningAxis/CommandTable

The following technology objects can be exported and imported as of TIA Portal \geq V16 for S7-300, S7-400, S7-1200, S7-1500:

- PID

Restrictions

The export format is internal and valid exclusively for the current version of TIA Portal Openness. The export format may change in future versions.

All errors which occur during the import and export are reported as exceptions.

For more information on exceptions, see section Handling exceptions (Page 740).

See also

[Field of application for Import/Export \(Page 750\)](#)

[Exporting configuration data \(Page 752\)](#)

8.1.2 Field of application for Import/Export

Introduction

The Import/Export functionality allows you to export specific objects in a targeted manner.

You can edit the exported data in an external program, or reuse it unchanged in other TIA Portal projects.

If you structure the import file correctly, you can also import configuration data created externally without having to carry out an export first.

Note

If you import externally created configuration data which contain code errors or a wrong structure this could cause unexpected errors.

Field of application

Exporting and importing data is useful for the following tasks:

- For externally editing configuration data.
- For importing externally-created configuration data, e.g. text lists and tags.
- For distributing specified configuration data to different projects, e.g. a modified process screen which is to be used in several projects.
- For replicating and adjusting the hardware configuration between the TIA Portal project and an ECAD program.

See also

[Basic principles of importing/exporting \(Page 747\)](#)

8.1.3 Version Specific Simatic ML Import

Application

As of TIA Portal Openness V14 SP1 the SimaticML import is useable cross-version. You will be able to import your older export files at least into the next two major versions.

Each version of the Openness API is able to import Simatic ML files from corresponding version and any supported version from previous release, for example import of Simatic ML file V14 SP1, V15 and V15.1 will be supported in Openness API V16.

The following table shows an example of which Simatic ML version can be imported by a given Openness API version.

Openness API version	Simatic ML file V14 SP1	Simatic ML file V15	Simatic ML V15.1	Simatic ML V16
V14 SP1	Import supported	Import unsupported	Import unsupported	Import unsupported
V15 SP1	Import supported	Import supported	Import unsupported	Import unsupported
V15.1	Import supported	Import supported	Import supported	Import unsupported
V16	Import supported	Import supported	Import supported	Import supported

Each version of the Openness API supports export of Simatic ML files. However, the version of the exported Simatic ML file should match with the version of the TIA Portal rather than that of the Openness API used.

To support this feature, SimaticML files contains the model version information as shown below:

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
  <Engineering version="V14 SP1"/>
  <DocumentInfo>
    ...
  </DocumentInfo>
  <SW.DataBlock ID="0">
    ...
  </SW.DataBlock>
</Document>
```

Note

If the version information is not provided in the SimaticML file, the system will use the current model version.

8.1.4 Editing the XML file

Introduction

Use an XML editor or text editor for editing an XML file for importing configuration data.

If you are making comprehensive changes or are creating custom object structures, we recommend that you use an XML editor with auto-complete function.

Note

Changing the XML content requires comprehensive knowledge of the structure and validation rules in XML. Work manually in the XML structure only in exceptional cases in order to avoid validation errors.

8.1.5 Exporting configuration data

Introduction

The configuration data of each start object (root) is exported separately to an XML file.

Editing the export file requires an adequate knowledge of XML. Use an XML editor for more convenient editing.

Example

You have a process screen that contains an IO field. An external tag is configured in the IO field. An export of the process screen includes the screen and the IO field. The tag and the connection used by the tag are not exported. Instead, only an open reference is included in the export.

Contents of the export file

Beginning with the start object, all objects of a sub-tree and their attributes are saved to the export file. All references to objects of different sub-trees are only exported as open references. The corresponding attributes of the referenced objects in different sub-trees are not written to the export file.

Note

Export of object types from the library is not supported

You can create objects as a type in the library. Instances of the object type used in the project can be edited like other objects using the TIA Portal Openness application. When you export objects, the instances are exported without the type information.

When you re-import these objects into the project, the instances of the object types are overwritten and the instance is detached from the object type.

The export file does not necessarily contain all the attributes of an object. You define what data is to be exported:

- ExportOptions.None
This setting exports only the modified data or the data that differs from the default.
The export file also contains all values that are obligatory for the subsequent data import.
- ExportOptions.WithDefaults¹
The default values are also exported.
- ExportOptions.WithReadOnly¹
The write-protected values are also exported.

¹: You can combine these two options with the following syntax:

```
Export(path, ExportOptions.WithDefaults |  
ExportOptions.WithReadOnly);
```

The entire contents of the export file are in English. Regardless of this, any project texts contained are exported and imported in all the languages present.

All configuration data is modeled as XML objects in the export file.

See also

[Basic principles of importing/exporting \(Page 747\)](#)

[Exporting blocks \(Page 868\)](#)

8.1.6 Importing configuration data

Introduction

Configuration data is imported from a previously exported and edited XML file or from an XML file you have created yourself. The data contained in this file is checked during the import. This approach prevents the configuration data in the TIA Portal from becoming inconsistent as a result of the import.

Restrictions

- All root objects in the import file have to be of the same kind, e.g. tag tables, blocks, ...
- If an import file includes several root objects and one of them is not valid, the entire contents of the import file are not imported.
- When importing texts, the corresponding project languages must have been set up in the target project in order to exclude import failure. If necessary you can modify the language settings via TIA Portal Openness.
- If you specify invalid attributes of an object in the import file that cannot be edited in the graphical user interface of TIA Portal, the import is canceled.
- Only the area pointers listed in the "separately for each connection" field can be imported or exported.

- The import of object types from the library is not supported. You can create objects as a type in the library. Instances of the object type used in the project can be edited like other objects using the TIA Portal Openness application. When you export objects, the instances are exported without the type information. When you re-import these objects into the project, the instances of the object types are overwritten and the instance is detached from the object type.
- The import of failsafe blocks is not supported.

Note

Device-dependent value ranges for graphical attributes

If values of graphical attributes exceed the valid value range, these values are reset to the possible maximum values for the HMI device during import.

Different import behavior

If objects to be imported already exist in the project, control the import behavior using different program codes. Otherwise, the objects will be created again in the project during the import.

The following settings for the import behavior are possible:

- `ImportOptions.None`
By using this setting configuration data will be imported without overwriting.
If an object being imported from an XML file which already exists in the project, the import is interrupted and an exception will be thrown.
- `ImportOptions.Override`
By using this setting configuration data will be imported with automatic overwriting.
You can specify that existing objects in the project are overwritten with the import. Relevant objects are deleted prior to the import and recreated with default values. These defaults are overwritten with the imported values during the import. If the existing object and the new object are not in the same group overwriting can't take place. To avoid naming conflicts import is canceled and an exception is thrown.

Procedure for importing

If you wish to import an XML file, the data it contains must adhere to certain rules. The contents of the import file must be well-formed. There must be no syntax errors and no data structure errors. In the case of comprehensive changes, use an XML editor that checks these criteria prior to the import.

During the import of the XML file to the TIA Portal, the data it contains is first checked for formal errors in the XML code. If errors are detected during the check, the import is canceled and the errors are shown in an exception (see Handling exceptions (Page 740)).

See also

[Basic principles of importing/exporting \(Page 747\)](#)

[Importing user data type \(Page 887\)](#)

8.2 Import/export of project data

8.2.1 Project graphics

8.2.1.1 Exporting/importing graphics

Introduction

The export of configuration data from the TIA Portal to the XML file does not include selected graphics, or graphics referenced by an object. The graphics are saved separately during the export. In the XML file, the graphics are referenced by a relative path and their file name. A graphic reference is modeled in the XML file as an object and contains an attribute list and, if necessary, a link list, just like the other objects.

```

<Hmi.Globalization.MultiLingualGraphic ID="0">
  <AttributeList>
    <DefaultDithering>False</DefaultDithering>
    <DefaultImageStream external="path">mygraphic files\DefaultImageStream.bmp</DefaultImageStream>
    <DefaultSmoothness>False</DefaultSmoothness>
    <Name>MyGraphic1</Name>
  </AttributeList>
  <ObjectList>
    <Hmi.Globalization.GraphicItem ID="1" CompositionName="Items">
      <AttributeList>
        <Culture>en-US</Culture>
        <Dithering>False</Dithering>
        <ImageStream external="path">mygraphic files\ImageStream.bmp</ImageStream>
        <Smoothness>False</Smoothness>
      </AttributeList>
    </Hmi.Globalization.GraphicItem>
    <Hmi.Globalization.GraphicItem ID="2" CompositionName="Items">
      <AttributeList>
        <Culture>de-DE</Culture>
        <Dithering>False</Dithering>
        <ImageStream external="path">mygraphic files\ImageStream_1.bmp</ImageStream>
        <Smoothness>False</Smoothness>
      </AttributeList>
    </Hmi.Globalization.GraphicItem>
  </ObjectList>
</Hmi.Globalization.MultiLingualGraphic>
```

Exporting graphics

The export of configuration data includes only graphics that were selected directly for export. The exportable graphics are stored in the TIA Portal for the specific language. If a project is configured in multiple languages, all the language versions used are exported.

When you export graphics, a new folder is created in the export file folder. The file folder name is built by concatenating the xml-filename with " files". This folder contains the exported

8.2 Import/export of project data

graphics. If this folder exists already, a new folder is created and supplemented by a consecutive number.

The graphics are saved in the same file format as in the project. The data format is not changed or converted, and the resolution and color depth remain unchanged.

The ID "default" is used as the file extension for the language selected as the default language.

If the folder already contains a file of the same name, the file name of the exported graphic is supplemented by a consecutive number.

Importing graphics

The following requirements apply when importing graphics:

- The graphics must have a file format that is supported by TIA Portal.
- The graphics must be referenced in the XML file by a relative path specification.

Once you have exported a graphic, you can edit it outside TIA Portal using a graphics program and then re-import it.

See also

[Basic principles of importing/exporting \(Page 747\)](#)

8.2.1.2 Exporting all graphics of a project

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
[See Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
[See Opening a project \(Page 106\)](#)

Application

You can export either a single graphic or all graphics of the graphics collection of a project in all languages. An XML file with all project graphic entries concerned is created during the export and referenced along with the exported graphics. The relevant graphics are saved along with the XML file to the same directory of the file system.

To allow the exported graphics ("*.jpg", "*.bmp", "*.png", "*.ico", etc.) to be changed, these graphics are not write-protected.

Program code: Exporting a graphic

Modify the following program code to export the required graphic:

```
//Exports all language variants of a single graphic
Project project = ...;
MultiLingualGraphicComposition graphicsComposition = project.Graphics;
MultiLingualGraphic graphic = graphicsComposition.Find("graphicName");
graphic.Export(new FileInfo(@"D:\ExportFolder\graphicName.xml"),
ExportOptions.WithDefaults);
```

Program code: Exporting all graphics

Modify the following program code to export all graphics of a graphics collection:

```
//Exports all graphics of a graphic library
Project project = ...;
MultiLingualGraphicComposition graphicsComposition = project.Graphics;
foreach(MultiLingualGraphic graphic in graphicsComposition)
{
    graphic.Export(new FileInfo(string.Format(@"D:\Graphics\{0}.xml", graphic.Name)),
ExportOptions.WithDefaults);
}
```

8.2.1.3 Importing graphics to a project

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

An XML file is saved along with the language versions of a graphic to a directory of your file system.

You can reference all graphics in a relative path in the XML file.

You can now import all language versions of a graphic contained in the XML file to the graphics collection.

You should also observe the Importing configuration data (Page 753).

Program code

Modify the following program code to import one or several graphics:

```
//Import all language variants of a single graphic  
Project project = ...;  
MultiLingualGraphicComposition graphicComposition = project.Graphics;  
graphicComposition.Import(new FileInfo(@"D:\Graphics\Graphic1.xml"),  
ImportOptions.Override);
```

8.2.2 Project texts

8.2.2.1 Export of project texts

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

In TIA Portal you can find project texts below the "Languages & resources" node of a project. These texts are exported to a "*.xlsx" file which is used for example for translations. The limitations of exporting and importing project texts are the same as in the UI. These limitations include:

- Exported texts can only be imported into the project from where they were exported.
- You can only translate texts to languages that are available in the project. If necessary you can add project languages via TIA Portal Openness.
- Only existing texts can be re-imported, if text from the original project has been deleted or re-created the import for that text will fail.

You have to define the following parameters:

Name	Example	Description
pah	new FileInfo ("D:\Test\Project-Text.xlsx")	Path to export file
sourceLanguage	new CultureInfo("en-US")	Reference language text is to be translated from
targetLanguage	new CultureInfo("de-DE")	Target language text is to be translated to

Note

Multilingual texts will be exported together with the parent object to which they belong.
Multilingual texts can not be exported explicitly.

Program code: Export from "Languages & resources" node

The use of the example parameters leads to the following program code to export project texts:

```
project.ExportProjectTexts(new FileInfo(@"D:\Test\ProjectText.xlsx"), new CultureInfo("en-US"), new CultureInfo("de-DE"));
```

XML structure of a exported multilingual text item

```
...
<MultilingualText ID="2" CompositionName="Comment">
  <ObjectList>
    <MultilingualTextItem ID="3" CompositionName="Items">
      <AttributeList>
        <Culture>en-US</Culture>
        <Text>My super tag</Text>
      </AttributeList>
    </MultilingualTextItem>
    <MultilingualTextItem ID="4" CompositionName="Items">
      <AttributeList>
        <Culture>ru-RU</Culture>
        <Text>Мой супер тэг</Text>
      </AttributeList>
    </MultilingualTextItem>
  </ObjectList>
</MultilingualText>
...

```

8.2.2.2 Import of project texts

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

In TIA Portal you can find project texts below the "Languages & resources" node of a project. You can import project texts from a ".xlsx" file which is used for example for translations. The limitations of exporting and importing project texts are the same as in the UI. These limitations include:

- Exported texts can only be imported into the project from where they were exported.
- You can only import translated texts in languages that are available in project from where they were exported.
- Only existing texts can be re-imported, if text from the original project has been deleted or re-created the import for that text will fail.

You have to define the following parameters:

Name	Example	Description
path	new FileInfo(@"D:\Test\Project-Text.xlsx")	Path to import file
updateSourceLanguage	true	If true, the text of the reference language is updated from the export file. If false, the text of the reference language is not updated

Note

Multilingual texts will be imported together with the parent object to which they belong.
Multilingual texts can not be imported explicitly.

Program code

The use of the example parameters leads to the following program code to import project texts:

```
ProjectTextResult result = project.ImportProjectTexts(new FileInfo(@"D:\Test\ProjectText.xlsx"), true);
```

The import of the Project Texts returns an object indicating the status of the Import and path to which the import log is saved. These attributes can be accessed with the following code:

```
ProjectTextResultState resultState = result.State;  
FileInfo logFilePath = result.Path;
```

8.3 Importing/exporting data of an HMI device

8.3.1 Structure of an XML file

Introduction

The data in the export file from the import/export is structured with reference to a basic structure.

Basic structure of an export file

The export file is generated in a XML format.

The XML file starts with a document information. It includes the data of the computer-specific installation with which the project was exported.

The export file is divided into the following two sections:

- Information about the document

In this section, you can enter your own information about the export in valid XML syntax. The content is ignored by the import.

For example you can insert a `<IntegrityInformation>...</IntegrityInformation>` block, in which you place additional information about the validation. After the XML file is forwarded, the recipient can use this block before the import to check whether the XML file has been changed.

- Object

This section contains the elements to be exported.

8.3 Importing/exporting data of an HMI device

```

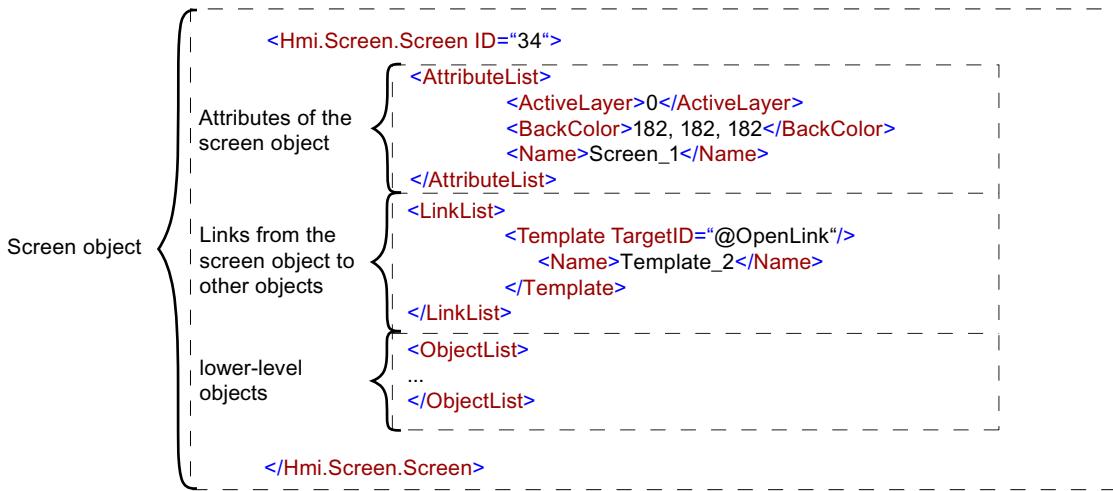
<?xml version="1.0" encoding="UTF-8" ?>
<Document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <DocumentInfo>
    <UserName>Jane Doe</UserName>
    <Company>Example_Inc</Company>
    <IntegrityInformation>...</IntegrityInformation>
    <Created>2016-04-28T18:05:42.179207Z</Created>
    <ExportSetting>WithDefaults</ExportSetting>
    <InstalledProducts>
      <Product>
        <DisplayName>Totally Integrated Automation Portal</DisplayName>
        <DisplayVersion>V14</DisplayVersion>
      </Product>
      <OptionPackage>
        <DisplayName>WinCC Professional</DisplayName>
        <DisplayVersion>V14</DisplayVersion>
      </OptionPackage>
      <OptionPackage>
        <DisplayName>Siemens TIA Openness</DisplayName>
        <DisplayVersion>V14</DisplayVersion>
      </OptionPackage>
    </InstalledProducts>
  </DocumentInfo>
  <Hmi.Screen.Screen ID ="0">
    <AttributeList>
      <ActiveLayer>0</ActiveLayer>
      <BackColor>189,190,0</BackColor>
      <Height>422</Height>
      <Name>Root screen</Name>
      <Number>1</Number>
      <Visible>True</Visible>
      <Width>640</Width>
    </AttributeList>
    <LinkList>
      <Template TargetID="@OpenLink">
        <Name>Template_1</Name>
      </Template>
    </LinkList>
    <ObjectList>
      <Name>dummy</Name>
    </ObjectList>
  </Hmi.Screen.Screen>
</Document>

```

The XML code represents the structure of an HMI device configuration. It includes information about the document (User Name, Company, Integrity Information, Creation Date, Export Settings) and a single screen object (Root screen). The Root screen is defined by its ID (0), Active Layer (0), Back Color (189,190,0), Height (422), Name (Root screen), Number (1), Visible status (True), and Width (640). It also contains a LinkList with one template named "Template_1" and an ObjectList containing a single object named "dummy". A brace on the left side of the code groups the "DocumentInfo" and "Hmi.Screen.Screen" sections under the heading "Information about the document". Another brace groups the entire "Hmi.Screen.Screen" section under the heading "Screen object".

Screen objects of an export file

The exported elements are available in additional elements of the XML file.



See also

[Basic principles of importing/exporting \(Page 747\)](#)

8.3.2 Structure of the data for importing/exporting

Objects

The basic structure is the same for all objects.

Every object in the XML file starts with its type, for example, "Hmi.Screen.Button", and an ID. The ID is created automatically during export.

```
<Hmi.Screen.Button CompositionName="ScreenItems" ID="60">
```

Each object apart from the start object also contains an "CompositionName" XML attribute. The value for this attribute is preset. It is occasionally necessary to specify this attribute, for example, to change the label when a button is pressed or released.

8.3 Importing/exporting data of an HMI device

```

<MultilingualText ID="A" CompositionName="TextOff">
  <ObjectList>
    <MultilingualTextItem ID="B" CompositionName="Items">
      <AttributeList>
        <Culture>en-US</Culture>
        <Text>
          <body>
            <p>TextOff</p>
          </body>
        </Text>
      </AttributeList>
    </MultilingualTextItem>
  </ObjectList>
</MultilingualText>
<MultilingualText ID="C" CompositionName="TextOn">
  <ObjectList>
    <MultilingualTextItem ID="D" CompositionName="Items">
      <AttributeList>
        <Culture>en-US</Culture>
        <Text>
          <body>
            <p>TextOn</p>
          </body>
        </Text>
      </AttributeList>
    </MultilingualTextItem>
  </ObjectList>
</MultilingualText>

```

Attributes

Every object contains attributes that are contained in an "AttributeList" section. Every attribute is modeled as an XML element, e.g. "BackColor". The value of an attribute is modeled as XML content, e.g. "204, 204, 204".

```

<Hmi.Screen.Button ID="2" CompositionName="ScreenItems">
  <AttributeList>
    <BackColor>204, 204, 204</BackColor>
    <ObjectName>Button_1</ObjectName>
  </AttributeList>
</Hmi.Screen.Button>

```

For referencing objects, each object contains a "LinkList" section, if necessary. This section contains links to other objects inside or outside the XML file. Every link is modeled as an XML element. The designation of a link is defined by the target object in the schema file. Every link also contains the "TargetID" attribute. When the target object is included in the XML file, the value of the "TargetID" attribute is the ID of the referenced object preceded by a "#". When the target object is not included in the XML file, the value of the "TargetID" attribute is "@OpenLink". The actual reference to the object is modeled as subordinate XML element.

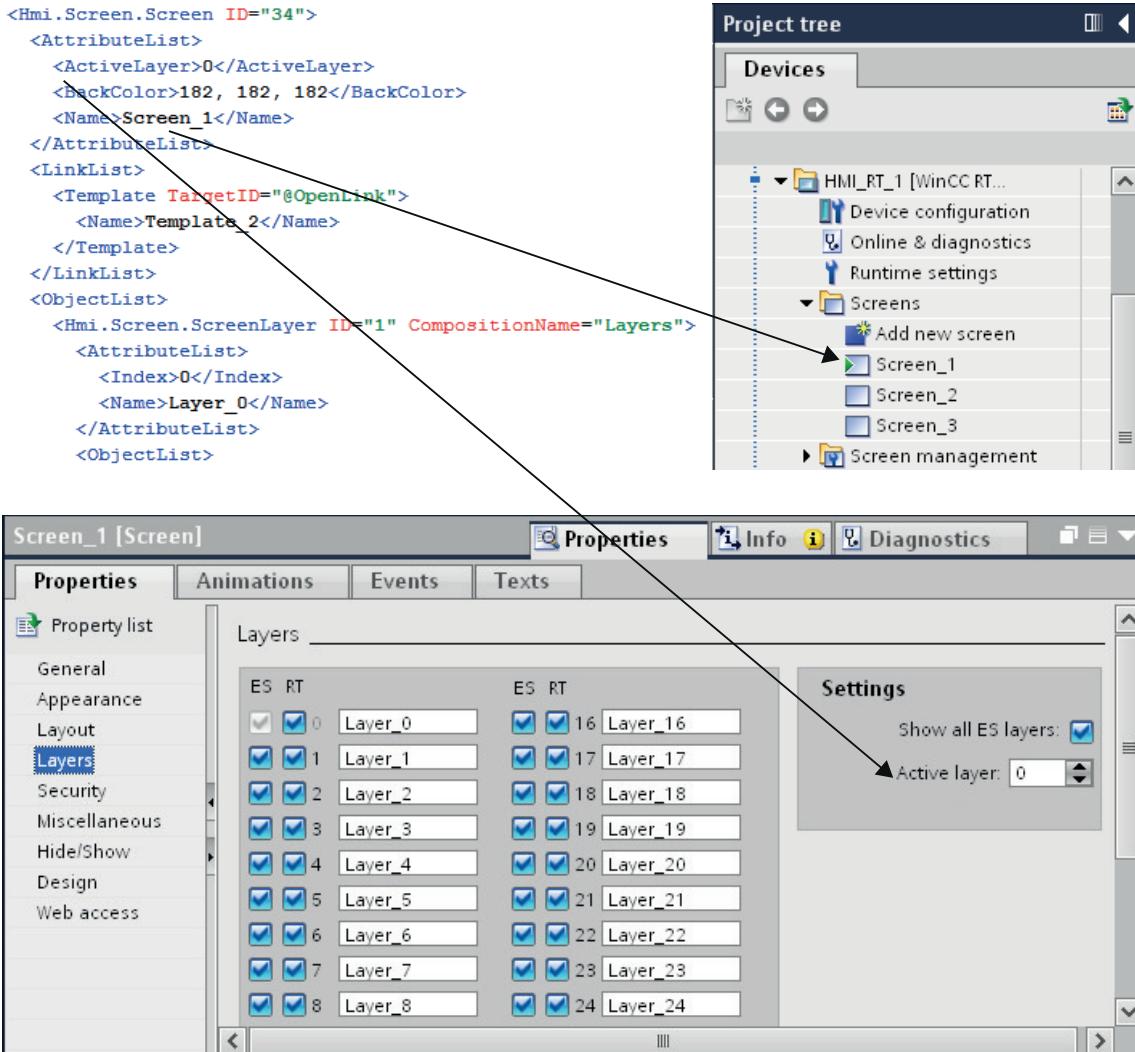
```

<Hmi.Tag.Tag ID="17">
    <AttributeList>
        <Name>Tag_1</Name>
    </AttributeList>
    <LinkList>
        <AcquisitionCycle TargetID="@OpenLink">
            <Name>2 s</Name>
        </AcquisitionCycle>
        <Connection TargetID="@OpenLink">
            <Name>HMI_connection</Name>
        </Connection>
    </LinkList>
</Hmi.Tag.Tag>

```

Relation between objects and XML structure

The figures below show the relation between the exported XML structure and the associated objects in WinCC.



8.3 Importing/exporting data of an HMI device

Figure 8-1 Relation between the WinCC user interface and the XML structure.

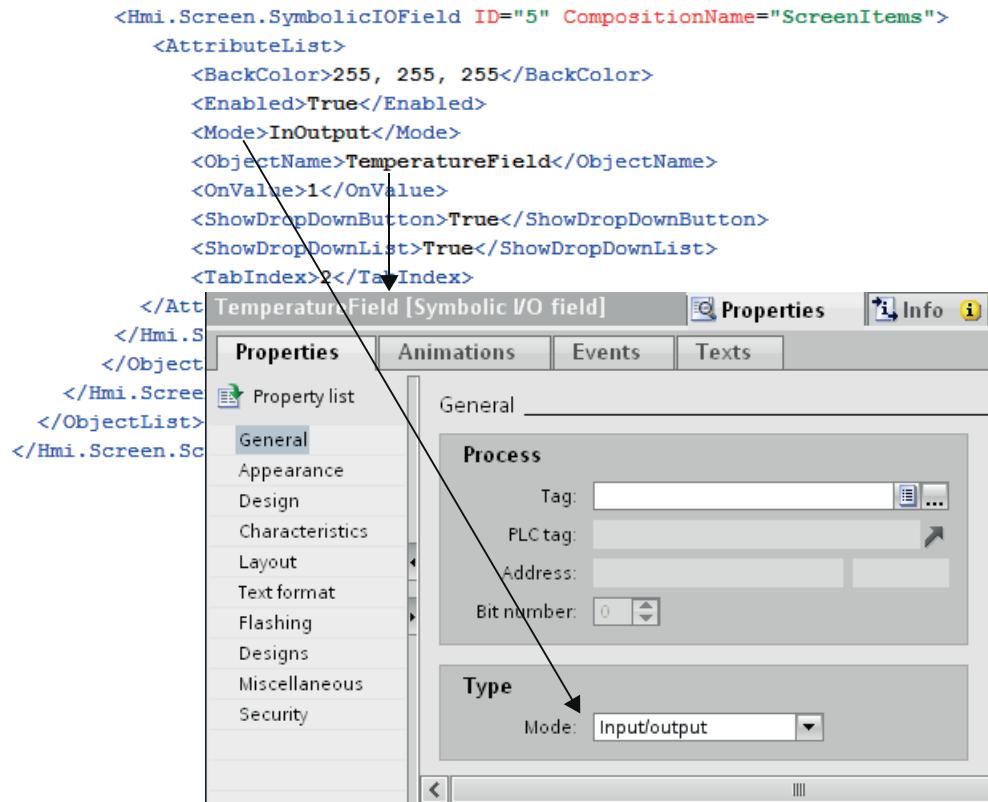


Figure 8-2 Relation between the settings in WinCC and the XML structure.

8.3.3 Cycles

8.3.3.1 Exporting cycles

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The TIA Portal Openness API interface supports the export of all cycles of a known HMI device to an XML file. The generation of the corresponding export file indicates that the export is complete.

Program code

Modify the following program code to export cycles from an HMI device to an XML file:

```
//Exports cycles from an HMI device
private static void ExportCyclesFromHMITarget(HmiTarget hmitarget)
{
    CycleComposition cycles = hmitarget.Cycles;
    foreach(Cycle cycle in cycles)
    {
        cycle.Export(new FileInfo(string.Format(@"C:\Samples\{0}.xml", cycle.Name)), ExportOptions.WithDefaults);
    }
}
```

8.3.3.2 Importing cycles

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

When you use `ImportOptions.None`, you can identify the cycles that have actually been imported based on the composition number (Composition count). You have access to these imported cycles.

Note

Standard cycles have attributes that cannot be edited in the user interface. If you specify in the import file that these attributes should be changed, the import causes a `NonrecoverableException` and closes the TIA Portal.

8.3 Importing/exporting data of an HMI device

Program code

Modify the following program code to import one or several cycles from an XML file into an HMI device:

```
//Imports cycles to an HMI device
private static void ImportCyclesToHMITarget(HmiTarget hmitarget)
{
    CycleComposition cycles = hmitarget.Cycles;
    string dirPathImport = @"C:\OpennessSamples\Import\";
    string cycleImportFileName = "CycleImport.xml";
    string fullFilePath = Path.Combine(dirPathImport, cycleImportFileName);

    cycles.Import(new FileInfo(fullFilePath), ImportOptions.None);
}
```

See also

[Importing configuration data \(Page 753\)](#)

8.3.4 Tag tables

8.3.4.1 Exporting HMI tag tables

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)

Application

One XML file is exported per HMI tag table. The API supports this export process. The export of tag tables is also available in subfolders.

Program code: Exporting all HMI tag tables from a specified folder

Modify the following program code to export all HMI tag tables from a specific folder:

```
//Exports all tag tables from a tag folder
private static void ExportAllTagTablesFromTagFolder(HmiTarget hmitarget)
{
    TagSystemFolder folder = hmitarget.TagFolder;
    TagTableComposition tables = folder.TagTables;

    foreach (TagTable table in tables)
    {
        FileInfo info = new FileInfo(string.Format(@"C:\OpennessSamples\TagTables\{0}.xml",
table.Name));
        table.Export(info, ExportOptions.WithDefaults);
    }
}
```

Program code: Exporting an HMI tag table

Modify the following program code to export an individual HMI tag table:

```
//Exports a tag table from an HMI device
private static void ExportTagTableFromHMITarget(HmiTarget hmitarget)
{
    string tableName = "Tag table XYZ";
    TagSystemFolder folder = hmitarget.TagFolder;
    TagTableComposition tables = folder.TagTables;
    TagTable table = tables.Find(tableName);

    if (table != null)
    {
        FileInfo info = new FileInfo(string.Format(@"C:\OpennessSamples\TagTables\{0}.xml",
table.Name));
        table.Export(info, ExportOptions.WithDefaults);
    }
}
```

Program code: Exporting all HMI tag tables

Modify the following program code to export all HMI tag tables:

```
//Exports all tag tables from an HMI device
private static void ExportAllTagTablesFromHMITarget(HmiTarget hmitarget)
{
    TagSystemFolder sysFolder = hmitarget.TagFolder;

    //First export the tables in underlying user folder
    foreach (TagUserFolder userFolder in sysFolder.Folders)
    {
        ExportUserFolderDeep(userFolder);
    }

    //then, export all tables in the system folder
    ExportTablesInSystemFolder(sysFolder);
}

private static void ExportUserFolderDeep(TagUserFolder rootUserFolder)
{
    foreach (TagUserFolder userFolder in rootUserFolder.Folders)
    {
        ExportUserFolderDeep(userFolder);
    }
    ExportTablesInUserFolder(rootUserFolder);
}

private static void ExportTablesInUserFolder(TagUserFolder folderToExport)
{
    TagTableComposition tables = folderToExport.TagTables;
    foreach (TagTable table in tables)
    {
        string fullFilePath = string.Format(@"C:\OpennessSamples\TagTables\{0}.xml",
table.Name);
        table.Export(new FileInfo(fullFilePath), ExportOptions.WithDefaults);
    }
}

private static void ExportTablesInSystemFolder(TagSystemFolder folderToExport)
{
    TagTableComposition tables = folderToExport.TagTables;
    foreach (TagTable table in tables)
    {
        string fullFilePath = string.Format(@"C:\OpennessSamples\TagTables\{0}.xml",
table.Name);
        table.Export(new FileInfo(fullFilePath), ExportOptions.WithDefaults);
    }
}
```

8.3.4.2 Importing HMI tag table

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Program code

Modify the following program code to import the HMI tag table of an XML file to a user-defined folder or to a system folder:

```
//Imports a single HMI tag table from a XML file
private static void ImportSingleHMITagTable(HmiTarget hmitarget)
{
    TagSystemFolder folder = hmitarget.TagFolder;
    TagTableComposition tables = folder.TagTables;

    FileInfo info = new FileInfo(@"D:\Samples\Import\myExportedTagTable.xml");
    tables.Import(info, ImportOptions.Override);
}
```

Incorrect import of tags

If you use the following symbols in the names of tags or referenced tags, the import of the tags will be faulty:

- . (period)
- \ (backslash)

Remedy 1:

Before an export, check that the name of the tag or referenced tag to be exported does not contain a period or backslash.

Remedy 2:

In the import file, exclude the names of tags or referenced tags using quotation marks.

Example

- Tag name with symbol:
`<name>Siemens.Simatic.Hmi.Utah.Tag.HmiTag:41000_Options_Time_Date
|DB_SFX0908_HMI1.Actual_Date_Time.Hour</name>`
- Tag name with symbol excluded in quotation marks:
`<name>"Siemens.Simatic.Hmi.Utah.Tag.HmiTag:41000_Options_Time_Date
|DB_SFX0908_HMI1.Actual_Date_Time.Hour"</name>`

8.3.4.3 Exporting an individual tag from an HMI tag table

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The following object model object types may possibly exist as sublevel items of an HMI tag and are taken into account during export:

MultilingualText	For Comment, TagValue, DisplayName
TagArrayMemberTag	For HMI array elements
TagStructureMember	For HMI structure elements
Event	For configured events
MultiplexEntry	For configured tag multiplexing entries

Program code

Modify the following program code to export an individual tag from an HMI tag table to an XML file:

```
//Exports a selected tag from a tag table
private static void ExportSelectedTagFromTagTable(HmiTarget hmitarget)
{
    TagSystemFolder tagFolder = hmitarget.TagFolder;
    TagTable mytable = tagFolder.TagTables.Find("MyTagTable");

    TagComposition containingTags = mytable.Tags;
    Tag myTag = containingTags.Find("MyTag");

    if (myTag != null)
    {
        FileInfo info = new FileInfo(string.Format(@"C:\OpennessSamples\Tags\{0}.xml",
myTag.Name));
        myTag.Export(info, ExportOptions.WithDefaults);
    }
}
```

8.3.4.4 Importing an individual tag into an HMI tag table

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The following object model object types may possibly exist as sublevel items of an HMI tag and are taken into account during import:

MultilingualText	For Comment, TagValue, DisplayName
TagArrayMemberTag	For HMI array elements
TagStructureMember	For HMI structure elements
Event	For configured events
MultiplexEntry	For configured tag multiplexing entries

Program code

Modify the following program code to import an HMI tag from an XML file into an HMI tag table:

```
//Imports a tag into a tag table
private static void ImportTagIntoTagTable(HmiTarget hmitarget)
{
    TagSystemFolder tagFolder = hmitarget.TagFolder;
    TagTable myTable = tagFolder.DefaultTagTable;
    TagComposition tagComposition = myTable.Tags;

    FileInfo info = new FileInfo(@"D:\Samples\Import\myExportedTag.xml");
    tagComposition.Import(info, ImportOptions.Override);
}
```

8.3.4.5 Special considerations for the export/import of HMI tags

Introduction

Special considerations apply to the export and import of the following HMI tags:

- External HMI tags with integrated connection
- HMI tags with the "UDT" data type

Similar program codes

The program code for the above-mentioned HMI tags is almost identical to the following program codes:

- Program code: Exporting HMI tags (Page 772)
- Program code: Importing HMI tags (Page 773)

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Special considerations for the export/import of an external HMI tag with integrated connection

When exporting an external HMI tag with integrated HMI connection, only the link of the HMI tag to the PLC tag is saved in the export file instead of the PLC tag data.

Before the import, you must ensure that the PLC, the corresponding PLC tags and the integrated connection to the corresponding PLC exist in the project. If this is not the case, these items must be created before the import. During the subsequent import of the external HMI tag, the link to the PLC tag will be activated again.

Names of external HMI tags must be unique across all tag tables of a project. If you do not specify the suitable tag table for the HMI tag during import, the import is canceled.

Use the following XML structure to import an external HMI tag with integrated connection:

```
<Hmi.Tag.Tag ID="1" CompositionName="Tags">
  <AttributeList>
    <Name>MyIntegratedHmiTag_1</Name>
  </AttributeList>
  <LinkList>
    <AcquisitionCycle TargetID="@OpenLink">
      <Name>1 s</Name>
    </AcquisitionCycle>
    <Connection TargetID="@OpenLink">
      <Name>HMI_Connection_MP277_300400</Name>      <- Must exist in the project
    </Connection>
    <ControllerTag TargetID="@OpenLink">
      <Name>Datablock_1.DBElement1</Name>          <- Must exist in the project
    </ControllerTag>
  </LinkList>
</Hmi.Tag.Tag>
```

Special considerations for the export/import of an HMI tag of the "UDT" data type

The link is exported to the data type when an HMI tag of the "UDT" data type is exported. Only versioned data types are supported for import.

The data types must be saved in the project library. Data types in the global library are not supported.

The following rules apply to the import:

- The referenced data type must be contained in the project library.
The import is terminated if the data type is not contained in the project library.
- The referenced data type must be versioned. Versioning is supported as of TIA Portal V13 SP1.
An exception is thrown if the data type is not versioned.

Note

The first data type found is used during the import to resolve the reference.

The following applies here: First, the root directory of the project library is searched, then the subfolders.

Use the following XML structure to import an HMI tag of the "UDT" data type:

```
<Hmi.Tag.Tag ID="1" CompositionName="Tags">
  ...
  <LinkList>
    <DataType TargetID="@OpenLink">
      <Name>HmiUdt_1 V 1.0.0</Name>           <- Must exist in the project library
    </DataType>
    ...
  </LinkList>
  ...
</Hmi.Tag.Tag>
```

8.3.5 VB scripts

8.3.5.1 Exporting VB scripts

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

All sublevel user-defined folders are taken into account for the export. A separate XML file is created for each exported VB script.

Program code: Exporting a VB script

Modify the following program code to export a selected VB script of an HMI device to an XML file:

```
//Exports a single vbscript of an HMI device
private static void ExportSingleVBScriptOfHMITarget(HmiTarget hmitarget)
{
    VBScriptSystemFolder vbScriptFolder = hmitarget.VBScriptFolder;
    VBScriptComposition vbScripts = vbScriptFolder.VBScripts;
    VBScript vbScript = vbScripts.Find("MyVBScript");

    FileInfo info = new FileInfo(string.Format(@"C:\OpennessSamples\Export\Scripts
\{0}.xml", vbScript.Name));
    vbScript.Export(info, ExportOptions.None);
}
```

8.3.5.2 Exporting VB scripts from a folder

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)

Application

A separate XML file is created for each exported VB script.

Program code: Exporting a VB script from a user-defined folder

Modify the following program code to export a VB script from a user-defined folder to an XML file:

```
//Exports vbscripts of a selected vbscript system folder
private static void ExportVBScriptOfSelectedVBScriptSystemFolder(HmiTarget hmitarget)
{
    VBScriptSystemFolder vbScriptFolder = hmitarget.VBScriptFolder;
    VBScriptUserFolderComposition vbUserFolders = vbScriptFolder.Folders;
    VBScriptUserFolder vbUserFolder = vbUserFolders.Find("MyVBUserFolder");
    VBScriptComposition vbScripts = vbUserFolder.VBScripts;

    foreach (VBScript script in vbScripts)
    {
        FileInfo info = new FileInfo(String.Format(@"C:\OpennessSamples\Export\Scripts
\{0}.xml", script.Name));
        script.Export(info, ExportOptions.None);
    }
}
```

Program code: Exporting all VB scripts from a system folder

Modify the following program code to export all VB scripts from the system folder:

```
//Exports all vbscripts by using a foreach loop
private static void ExportAllVBScripts(HmiTarget hmitarget)
{
    VBScriptSystemFolder vbScriptFolder = hmitarget.VBScriptFolder;
    VBScriptComposition vbScripts = vbScriptFolder.VBScripts;
    if (vbScripts == null) return;

    foreach (VBScript script in vbScripts)
    {
        FileInfo info = new FileInfo(string.Format(@"C:\OpennessSamples\Export\Scripts
\{0}.xml", script.Name));
        script.Export(info, ExportOptions.None);
    }
}
```

8.3.5.3 Importing VB scripts

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

8.3 Importing/exporting data of an HMI device

Application

Bulk imports are supported. As an alternative, you can use a program code with a Foreach loop (Exporting VB scripts (Page 775)).

Program code

Modify the following program code to import a VB script from an XML file into an HMI device:

```
private static void ImportSingleVBScriptToHMITarget(HmiTarget hmitarget)
{
    VBScriptSystemFolder vbScriptFolder = hmitarget.VBScriptFolder;
    VBScriptComposition vbScripts = vbScriptFolder.VBScripts;
    if (vbScripts == null) return;
    {
        FileInfo info = new FileInfo(@"D:\Samples\Import\VBScript.xml");
        vbScripts.Import(info, ImportOptions.None);
    }
}
```

8.3.6 Text lists

8.3.6.1 Exporting text lists from an HMI device

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)

Application

The export of text and graphic lists includes all their entries. Text and graphic lists can be exported separately.

The text lists of an HMI device are exported. A separate XML file is created for each exported text list.

Program code

Modify the following program code to export text lists from an HMI device:

```
//Export TextLists
private static void ExportTextLists(HmiTarget hmitarget)
{
    TextListComposition text = hmitarget.TextLists;
    foreach (TextList textList in text)
    {
        FileInfo info = new FileInfo(string.Format(@"D:\Samples\Export\{0}.xml",
textList.Name);
        textList.Export(info, ExportOptions.WithDefaults);
    }
}
```

8.3.6.2 Importing a text list into an HMI device

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The API interface supports the import of a text list from an XML file into an HMI device.

Program code

Modify the following program code to import a text list from an XML file into an HMI device:

```
//Imports a single TextList
private static void ImportSingleTextList(HmiTarget hmitarget)
{
    TextListComposition textListComposition = hmitarget.TextLists;
    IList<TextList> importedTextLists = textListComposition.Import(new
FileInfo(@"D:\SamplesImport\myTextList.xml"), ImportOptions.Override);
}
```

8.3.6.3 Advanced XML formats for export/import of text lists

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- Standard export of text lists
See Exporting text lists from an HMI device (Page 778)
- Standard import of text lists
See Importing text lists into an HMI device (Page 779)

Application

A text list may also contain formatted texts. This primarily concerns the following formatting:

- Text formatting
- References to other objects within the text

Pure text formatting within a text list to be exported results in an advanced XML export format. Object references are characterized as Open Links. The same applies to text lists to be imported with formatted texts.

Advanced XML export formats may also become considerably more complex. For example, more than just the object name may be linked in the text list, perhaps by means of an Open Link to a PLC tag of a different device. In this case, all information must be coded in a string in order to remove the Open Link.

```

<?xml version="1.0" encoding="utf-8"?>
<Document>
<!-- ... -->
<MultilingualText ID="5" CompositionName="Text">
    <ObjectList>
        <MultilingualTextItem ID="6" CompositionName="Items">
            <AttributeList>
                <Culture>en-US</Culture>
                <Text>
                    <body>
                        <p>
                            <field ref="0" />
                        </p>
                    </body>
                <fieldinfos>
                    <fieldinfo name="0" domaintype="HMICommonTextList">
                        <reference TargetID="@OpenLink">
                            <name>Siemens.Simatic.Hmi.Utah.TextAndGraphicLists.HmiTextList:Empty Text_li
                        </reference>
                        <subreference TargetID="@OpenLink">
                            <name>Siemens.Simatic.Hmi.Utah.Tag.HmiTag:t1</name>
                        </subreference>
                        <domaindata>
                            <format length="9" />
                        </domaindata>
                    </fieldinfo>
                </fieldinfos>
            </Text>
        </AttributeList>
    </MultilingualTextItem>
    <MultilingualTextItem ID="7" CompositionName="Items">
        <AttributeList>
            <Culture>de-CH</Culture>
            <Text>
                <body>
                    <p>
                        <field ref="0" />
                    </p>
                </body>
            <fieldinfos>
                <fieldinfo name="0" domaintype="HMICommonTextList">
                    <reference TargetID="@OpenLink">
                        <name>Siemens.Simatic.Hmi.Utah.TextAndGraphicLists.HmiTextList:Empty Text_li
                    </reference>
                    <subreference TargetID="@OpenLink">
                        <name>Siemens.Simatic.Hmi.Utah.Tag.HmiTag:t1</name>
                    </subreference>
                    <domaindata>
                        <format length="9" />
                    </domaindata>
                </fieldinfo>
            </fieldinfos>
        </Text>
    </AttributeList>
    </MultilingualTextItem>
    </ObjectList>
</MultilingualText>

```

8.3.7 Graphic lists

8.3.7.1 Exporting graphic lists

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The export of text and graphic lists includes all their entries. Text and graphic lists can be exported separately.

One XML file is created per graphic list. Global graphic objects contained in the graphic lists are exported as Open Links.

Program code

Modify the following program code to export graphic lists of an HMI device:

```
//Exports GraphicLists
private static void ExportGraphicLists(HmiTarget hmitarget)
{
    GraphicListComposition graphic = hmitarget.GraphicLists;
    foreach (GraphicList graphicList in graphic)
    {
        FileInfo info = new FileInfo(string.Format(@"D:\Samples\Export\{0}.xml",
graphicList.Name));
        graphicList.Export(info, ExportOptions.WithDefaults);
    }
}
```

8.3.7.2 Importing a graphic list

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The API interface supports the import of a graphic list from an XML file into an HMI device.

All referenced graphic objects of the graphic list are included in the import. References to global graphics are not included. If the referenced global graphics exist in the target project, the references to the global graphics are restored during the import.

Program code

Modify the following program code to import a graphic list from an XML file into an HMI device:

```
//Imports a single GraphicList
private static void ImportSingleGraphicList(HmiTarget hmitarget)
{
    GraphicListComposition graphicListComposition = hmitarget.GraphicLists;
    IList<GraphicList> importedGraphicLists = graphicListComposition.Import(new
    FileInfo(@"D:\Samples\Import\myGraphicList.xml"), ImportOptions.Override);
}
```

8.3.8 Connections

8.3.8.1 Exporting connections

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The API interface supports the export of all connections of an HMI device to an XML file.

Note

Export of integrated connections

Export of integrated connections is not supported.

A separate XML file is created for each exported connection.

8.3 Importing/exporting data of an HMI device

Program code

Modify the following program code to export all connections of an HMI device to an XML file:

```
//Exports communication connections from an HMI device
private static void ExportConnectionsFromHMITarget(HmiTarget hmitarget)
{
    ConnectionComposition connections = hmitarget.Connections;
    foreach(Connection connection in connections)
    {
        FileInfo info = new FileInfo(string.Format(@"D:\Samples\Export\{0}.xml",
connection.Name));
        connection.Export(info, ExportOptions.WithDefaults);
    }
}
```

8.3.8.2 Importing connections

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The API interface supports the import of all connections of an HMI device from an XML file into an HMI device. If you want to import several communication connections, import the corresponding XML file for each one.

Note

If you import a connection into a project in which an integrated connection has already been configured, this connection is not overwritten. The import is canceled and an Exception is thrown.

Program code

Modify the following program code to import an individual connection of an HMI device from an XML file into an HMI device:

```
//Imports Communication connections to an HMI device
private static void ImportConnectionsToHMITarget(HmiTarget hmitarget)
{
    ConnectionComposition connections = hmitarget.Connections;
    IList<Connection> importedConnectionLists = connections.Import(new
FileInfo(@"D:\Samples\Import\myConnectionImport.xml"), ImportOptions.Override);
}
```

8.3.9 Screens

8.3.9.1 Overview of exportable screen objects

Application

You can export or import the screens below using TIA Portal Openness APIs:

Table 8-4 Supported screens

Object	Export/import possible
Screen	Yes
Global screen	Yes
Screen template	Yes
Permanent area	Yes
Pop-up screen	Yes
Slide-in screen	Yes

You can export or import the screen objects below using TIA Portal Openness APIs:

Table 8-5 Supported screen objects

Range	Object type	Export/import possible
Basic objects	Line	Yes
	Polyline	Yes
	Polygon	Yes
	Ellipse	Yes
	Ellipse segment	–
	Circle segment	–
	Elliptical arc	–
	Circular arc	–
	Circle	Yes
	Rectangle	Yes
	Connector	–
	Text field	Yes
	Graphic view	Yes
	Pipe	–
	Double T-piece	–
	T-piece	–
	Pipe elbow	–

Range	Object type	Export/import possible
Elements	I/O field	Yes
	Graphic I/O field	Yes
	Editable text field	–
	List box	–
	Combo box	–
	Button	Yes
	Round button	–
	Illuminated button	Yes
	Switch	Yes
	Symbolic I/O field	Yes
	Date/time field	Yes
	Bar	Yes
	Symbol library	Yes
	Slider	Yes
	Scroll bar	–
	Check box	–
	Option buttons	–
	Gauge	Yes
	Clock	Yes
	Memory space view	–
	Function keys (softkeys)	Yes
	Groups	Yes
	Faceplate instances	Yes

Range	Object type	Export/import possible
Controls	Screen window	–
	User view	Yes
	Print job/script diagnostics	–
	Camera view	–
	PDF view	–
	Recipe view	–
	Alarm view	–
	Alarm indicator	–
	Alarm window	–
	f(x) trend view	–
	f(t) trend view	–
	Table view	–
	Value table	–
	HTML Browser	–
	Media Player	–
	Channel diagnostics	–
	WLAN reception	–
	Zone name	–
	Zone signal	–
	Effective range name	–
	Effective range name (RFID)	–
	Effective range signal	–
	Charge condition	–
	Handwheel	–
	Help indicator	–
	Sm@rtClient view	–
	Status/Force	–
	Memory space view	–
	NC subprogram display	–
	System diagnostic view	–
	System diagnostic window	–

See also

[Basic principles of importing/exporting \(Page 747\)](#)

8.3.9.2 Exporting all screens of an HMI device

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

A different program code is required for exporting all aggregated screens of all user-defined screen folders of an HMI device.

Program code: Exporting all screens of a device

Modify the following program code to export the screens of a user-defined screen folder of an HMI device and the screen system folder:

```
private static void ExportScreensOfDayce(string rootPath, HmiTarget hmitarget)
{
    DirectoryInfo info = new DirectoryInfo(rootPath);
    info.Create();
    //export the ScreenFolder recursive

    string screenPath = Path.Combine(rootPath, "Screens");
    info = new DirectoryInfo(screenPath);
    info.Create();
    ExportScreens(screenPath, hmitarget);
}
```

Program code: Exporting all screens of a user-defined folder

Modify the following program code to export the screens of a user-defined screen folder of an HMI device and the screen system folder:

```
private static void ExportScreensOfDayce(HmiTarget hmitarget)
{
    ScreenUserFolder folder = hmitarget.ScreenFolder.Folders.Find("MyScreenFolder");
    //or ScreenSystemFolder folder = hmitarget.ScreenFolder;
    ScreenComposition screens = folder.Screens;
    foreach(Screen screen in screens)
    {
        FileInfo info = new FileInfo(string.Format(@"D:\Samples\Screens\{0}\{1}.xml",
        folder.Name, screen.Name));
        screen.Export(info, ExportOptions.WithDefaults);
    }
}
```

Program code: Exporting all screens of a device independent of the user

Modify the following program code to export all screens:

```
public static void ExportScreens(string screenPath, HmiTarget target)
{
    foreach(Screen screen in target.ScreenFolder.Screens)
    {
        screen.Export(new FileInfo(Path.Combine(screenPath, screen.Name + ".xml")),
ExportOptions.WithDefaults);
    }
    foreach(ScreenUserFolder subfolder in target.ScreenFolder.Folders)
    {
        ExportScreenUserFolder(Path.Combine(screenPath, folder.Name), subfolder);
    }
}

private static void ExportScreenUserFolder(string screenPath, ScreenUserFolder folder )
{
    foreach(Screen screen in folder.Screens)
    {
        screen.Export(new FileInfo(Path.Combine(screenPath, screen.Name + ".xml")),
ExportOptions.WithDefaults);
    }
    foreach(ScreenUserFolder subfolder in folder.Folders)
    {
        ExportScreenUserFolder(Path.Combine(screenPath, subfolder.Name), subfolder);
    }
}
```

8.3.9.3 Exporting a screen from a screen folder

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The following data of a screen is exported:

Screen	Data
Attributes	ActiveLayer, BackColor, Height, Width, Name, Number, HelpText
Open links	Template
Compositions	<ul style="list-style-type: none"> • Layers • Animations All configured animations that are based on Runtime Advanced are exported. • Events All configured events that are based on Runtime Advanced are exported. • Softkeys All configured softkeys are exported.

The following data is exported for each layer:

Note

By default, the layer name in the TIA Portal is an empty text.

If you do not change the layer name in the TIA Portal, the exported layer name will be an empty text. In this case, the displayed layer name in the TIA Portal depends on the user interface language.

If you do change the layer name in the TIA Portal, the modified layer name is displayed in all relevant languages.

Layer	Data
Attributes	Name, Index, VisibleES
Compositions	ScreenItems (with screen items)

Not included in the export:

- SCADA-specific attributes.
- Layers that do not contain any screen items and whose attributes do not differ from the default values.

8.3 Importing/exporting data of an HMI device

Program code

Modify the following program code to export an individual screen from the user folder or from the system folder of an HMI device:

```
//Exports a single screen from a screen folder
private static void ExportSingleScreenFromScreenFolder(HmiTarget hmitarget)
{
    ScreenUserFolder folder = hmitarget.ScreenFolder.Folders.Find("MyScreenFolder");
    //or ScreenSystemFolder folder = hmitarget.ScreenFolder;
    ScreenComposition screens = folder.Screens;
    Screen screen = screens.Find("Screen_1.xml");
    if (screen == null) return;
    {
        FileInfo info = new FileInfo(string.Format(@"D:\Samples\Screens\{0}\{1}.xml",
        folder.Name, screen.Name));
        screen.Export(info, ExportOptions.WithDefaults);
    }
}
```

8.3.9.4 Importing screens to an HMI device

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The screens can only be imported to a specific type of HMI device. The HMI device and the device from which the screens were exported must be of the same device type.

The following data of a screen is imported:

Screen	Data
Attributes	ActiveLayer, BackColor, Height, Width, Name, Number, HelpText
Open links	Templates
Compositions	<ul style="list-style-type: none">• Layers• Animations All animations configurable for screens are imported.• Events All events configurable for screens are imported.• Softkeys All softkeys configurable for screens are imported.

The following data is imported for each layer:

Note

If you have specified an empty text for the layer name before the import, the displayed layer name in the TIA Portal after the import depends on the user interface language.

If you have assigned a layer name, the specified layer name is displayed in all relevant languages after the import.

Layer	Data
Attributes	Name, Index
Compositions	ScreenItems

Restrictions

- The import is canceled and an Exception is thrown if the width and height of a screen do not correspond to the dimensions of the device. Adaptation of the screen items contained is not supported. For this reason, certain screen items may be located beyond the screen boundaries. A compiler warning is output in this case.
- The screen number must be unique for all screens of the device. A screen import is canceled if a screen with a screen number that was already created in the device is found. If you have not yet assigned a screen number, a unique number is assigned to the screen during the import.
- The layout of the screen items within the Z-order must be unique and contiguous for each layer in the screen. For this reason, after the import of the screen, a consistency check is performed that repairs the layout, if necessary. This action may lead to modified "tab indexes" for certain screen items.

You can change the Z-order of the screen items in the XML file manually. The screen item in first place is at the very end in the Z-order.

Note

You can change the values for width and height of the screen items in the XML file if the attribute "Fit size to content" is enabled for the screen item.

Note

Import of screen types from the library is not supported

As of WinCC V12 SP1, you can create a screen as type in the library. Instances of the screen type used in the project can be edited like other screens using the TIA Portal Openness application. When you export screens, the instances of screen types are exported without the type information.

When you re-import these screens into the project, the instances of the screen types are overwritten and the instance is detached from the screen type.

Program code: Importing screens to an HMI device

Modify the following program code to import screens to an HMI device using a For each loop:

```
//Imports all screens to an HMI device
private static void ImportScreensToHMITarget(HmiTarget hmitarget)
{
    FileInfo[] exportedScreens = new FileInfo[] {new FileInfo(@"D:\Samples\Import\Screen_1.xml"), new FileInfo(@"D:\Samples\Import\Screen_2.xml")};
    ScreenUserFolder folder = hmitarget.ScreenFolder.Folders.Find("MyScreenFolder");
    foreach (FileInfo screenFileInfo in exportedScreens)
    {
        folder.Screens.Import(screenFileInfo, ImportOptions.Override);
    }
}
```

Program code: Import to a newly created user folder

Modify the following program code to import a screen to a newly created user folder of an HMI device:

```
//Imports a single screen to a new created user folder of an HMI device
private static void ImportSingleScreenToNewFolderOfHMITarget(HmiTarget hmitarget)
{
    ScreenUserFolder folder = hmitarget.ScreenFolder.Folders.Create("MyFolder");
    folder.Screens.Import(new FileInfo(@"D:\Samples\Import\myScreens.xml"),
ImportOptions.Override);
}
```

8.3.9.5 Exporting permanent areas

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)

Application

The following data of the permanent area is exported:

Permanent area	Data
Attributes	ActiveLayer, BackColor, Height, Width, Name
Compositions	Layers

The following data is exported for each layer:

Layer	Data
Attributes	Name, Index
Compositions	ScreenItems (with screen items)

Program code

Modify the following program code to export a permanent area of an HMI device to an XML file:

```
//Exports a permanent area
private static void ExportScreenoverview(HmiTarget hmitarget)
{
    ScreenOverview overview = hmitarget.ScreenOverview;
    if (overview == null) return;

    FileInfo info = new FileInfo(@"D:\Samples\Screens\ExportedOverview.xml");
    overview.Export(info, ExportOptions.WithDefaults);
}
```

8.3.9.6 Importing permanent areas

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)

Application

The following data of the permanent area is imported:

Permanent area	Data
Attributes	ActiveLayer, BackColor, Height, Width, Name, Visible, Number
Compositions	Layers

The following data is imported for each layer:

Layer	Data
Attributes	Name, Index
Compositions	ScreenItems (with screen items)

The import is canceled and an Exception is thrown if the width and height of a screen do not correspond to the dimensions of the device. Adaptation of the included device items (Screen items) is not supported. For this reason, some device items may be located beyond the screen boundaries. A compiler warning is output in this case.

8.3 Importing/exporting data of an HMI device

The layout of the device items must be unique and contiguous in the permanent area. For this reason, after the import of the permanent area, a consistency check is performed that repairs the layout, if necessary. This action may lead to modified "tab indexes" for certain device items.

Program code

Modify the following program code to import a permanent area from an XML file into an HMI device:

```
//Imports a permanent area
private static void ImportScreenOverview(HmiTarget hmiTarget)
{
    FileInfo info = new FileInfo(@"D:\Samples\Screens\ExportedOverview.xml");
    hmiTarget.ImportScreenOverview(info, ImportOptions.Override);
}
```

8.3.9.7 Exporting all screen templates of an HMI device

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)

Application

One XML file is created per screen template.

Because bulk exports are not supported, you need to enumerate and export all screen templates separately. In the course of this action, make sure that the screen template names used conform to the file naming conventions of your file system.

Program code: Exporting all screen templates of a device

Modify the following program code to export all screen templates from a specific folder:

```
public static void ExportScreenTemplatesOfDevice(string rootPath ,
ScreenTemplateUserFolder folder)
{
    string screenPath = Path.Combine(rootPath, "Screens");
    DirectoryInfo info = new DirectoryInfo(screenPath);
    info.Create();

    //export the ScreenTemplateFolder recursive
    ExportScreenTemplates (screenPath, hmitarget);
}
```

Program code: Exporting all screen templates of a specific folder

Modify the following program code to export all screen templates:

```
//Exports all screen templates of a selected folder
private static void ExportScreenTemplates(string templatePath, HmiTarget hmitarget)
{
    foreach (ScreenTemplate screen in hmitarget.ScreenTemplateFolder.ScreenTemplates)
    {
        screen.Export(new FileInfo(Path.Combine(templatePath, screen.Name + ".xml")),
ExportOptions.WithDefaults);
    }
    foreach (ScreenTemplateUserFolder folder in hmitarget.ScreenTemplateFolder.Folders)
    {
        ExportScreenTemplates(Path.Combine(templatePath, folder.Name), hmitarget);
    }
}
```

8.3.9.8 Exporting screen templates from a folder

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The following data of the screen template is exported:

Screen templates	Data
Attributes	ActiveLayer, BackColor, Height, Width, Name
Compositions	<ul style="list-style-type: none"> • Layers • Animations All configured animations are exported. SCADA animations are not exported. • Softkeys All configured softkeys are exported.

The following data is exported for each layer:

Layer	Data
Attributes	Name, Index
Compositions	ScreenItems (with screen items)

Program code: Exporting a screen template of a user-defined folder

Modify the following program code to export an individual screen template from the system folder or from a user-defined folder:

```
private static void ExportSingleScreenTemplate(string templatePath, HmiTarget hmiTarget)
{
    ScreenTemplateUserFolder folder =
hmiTarget.ScreenTemplateFolder.Folders.Find("MyTemplateFolder");
    //or ScreenTemplateSystemFolder folder = hmiTarget.ScreenTemplateFolder;
    ScreenTemplateComposition templates = folder.ScreenTemplates;
    ScreenTemplate template = templates.Find("templateName");
    if(template == null) return;

    FileInfo info = new FileInfo(string.Format(@"D:\Samples\Templates\{0}\{1}.xml",
folder.Name, template.Name));
    template.Export(info, ExportOptions.WithDefaults);
}
```

Program code: Exporting all screen templates of a user-defined folder

Modify the following program code to export all screen templates from a specific folder:

```
public static void ExportScreenTemplateUserFolder(string rootPath,
ScreenTemplateUserFolder folder)
{
    DirectoryInfo info = new DirectoryInfo(rootPath);
    info.Create();

    foreach (ScreenTemplate screen in folder.ScreenTemplates)
    {
        screen.Export(new FileInfo(Path.Combine(info.FullName, screen.Name + ".xml")),
ExportOptions.WithDefaults);
    }
    foreach (ScreenTemplateUserFolder subfolder in folder.Folders)
    {
        ExportScreenTemplateUserFolder(Path.Combine(info.FullName, subfolder.Name),
subfolder);
    }
}
```

Program code: Exporting all screen templates of a specific folder

Modify the following program code to export all screen templates:

```
//Exports all screen templates of a selected folder
private static void ExportScreenTemplates(string templatePath, ScreenTemplateUserFolder
folder)
{
    foreach (ScreenTemplate screen in folder.ScreenTemplates)
    {
        screen.Export(new FileInfo(Path.Combine(templatePath, screen.Name + ".xml")), 
ExportOptions.WithDefaults);
    }
    foreach (ScreenTemplateUserFolder subfolder in folders.Folders)
    {
        ExportScreenTemplates(Path.Combine(templatePath, subfolder.Name), subfolder);
    }
}
```

8.3.9.9 Importing screen templates

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The following data of a screen template is imported:

Screen template	Data
Attributes	ActiveLayer, BackColor, Height, Width, Name, SetTabOrderInFront
Compositions	<ul style="list-style-type: none"> • Layers • Animations All animations configurable for screens are imported. • Softkeys All softkeys configurable for screens are imported.

The following data is imported for each layer:

Layer	Data
Attributes	Name, Index
Compositions	ScreenItems (with screen items)

The import is canceled and an Exception is thrown if the width and height of a screen template do not correspond to the dimensions of the device. Adaptation of the included screen items is

8.3 Importing/exporting data of an HMI device

not supported. For this reason, certain screen items may be located beyond the screen boundaries. A compiler warning is output in this case.

The layout of the screen items must be unique and contiguous in the screen template. For this reason, after the import of the screen template, a consistency check is performed which repairs the layout, if necessary. This action may lead to modified "tab indexes" for certain screen items.

Program code: General import

Modify the following program code to import all screen templates to an HMI device using a For each loop:

```
//Imports screen templates to an HMI device
private static void ImportScreenTemplatesToHMITarget(HmiTarget hmitarget)
{
    ScreenTemplateUserFolder folder =
hmitarget.ScreenTemplateFolder.Folders.Find("MyTemplateFolder");
    // or ScreenTemplateSystemFolder folder = hmitarget.ScreenTemplateFolder;
    FileInfo[] exportedTemplates = {new FileInfo[] { new FileInfo(@"D:\Samples\Import
\Template_1.xml"), new FileInfo(@"D:\Samples\Import\Template_n.xml") } };
    foreach (FileInfo templateFileName in exportedTemplates)
    {
        folder.ScreenTemplates.Import(templateFileName, ImportOptions.Override);
    }
}
```

Program code: Import into a newly created user folder

Modify the following program code to import a screen template into a newly created user folder of an HMI device:

```
//Imports screen templates to a user folder of an HMI device
private static void ImportScreenTemplatesToFOLDEROfHMITarget(HmiTarget hmitarget)
{
    ScreenTemplateUserFolder screenTemplateFolder =
hmitarget.ScreenTemplateFolder.Folders.Find("MyTemplateFolder");
    ScreenTemplateUserFolder folder = screenTemplateFolder.Folders.Create("MyNewFolder");
    folder.ScreenTemplates.Import(new FileInfo(@"D:\Samples\Import\ScreenTemplate.xml"),
ImportOptions.Override);
}
```

8.3.9.10 Exporting a pop-up screen

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The following data of the pop-up screen is exported:

Screen templates	Data
Attributes	ActiveLayer, BackColor, GridColor, Height, Name, ScrollbarBackgroundColor, ScrollbarForegroundColor, Width
Compositions	<ul style="list-style-type: none"> • Layers • Events All configured events are exported.

The following data is exported for each layer:

Layer	Data
Attributes	Name, Index, VisibleES
Compositions	ScreenItems All exportable screen objects are exported.

Program code: Exporting a pop-up screen from a folder

Modify the following program code to export an individual pop-up screen from the system folder or from a user-defined folder:

```
//Exports a single pop-up screen
private static void ExportSinglePopUpScreen(HmiTarget hmitarget)
{
    ScreenPopupUserFolder folder =
    hmitarget.ScreenPopupFolder.Folders.Find("MyPopupFolder");
    //or ScreenPopupSystemFolder folder = hmitarget.ScreenPopupFolder;
    ScreenPopupComposition popups = folder.ScreenPopups;
    ScreenPopup popup = popups.Find("popupName");
    if(popup == null) return;

    FileInfo info = new FileInfo(string.Format(@"D:\Samples\Screens\{0}\{1}.xml",
    folder.Name, popup.Name);
    popup.Export(info, ExportOptions.WithDefaults);
}
```

8.3.9.11 Importing a pop-up screen

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The following data of a pop-up screen is imported:

Screen templates	Data
Attributes	ActiveLayer, BackColor, GridColor, Height, Name, ScrollbarBackgroundColor, ScrollbarForegroundColor, Width
Compositions	<ul style="list-style-type: none">• Layers• Events <p>All configured events are exported.</p>

The existence of the following attributes is mandatory for the import:

- Name
- Height
- Width

The following data is imported for each layer:

Layer	Data
Attributes	Name, Index, VisibleES
Compositions	ScreenItems All importable screen objects are imported.

Restrictions

If a device doesn't support pop-up screens, the import is cancelled and an Exception is thrown.

If the width and height of a pop-up screen do not comply to the following dimensions restrictions for a device, the import is cancelled and an Exception is thrown:

- Minimum height = 1 pixel
- Minimum width = 1 pixel
- Maximum height = sixfold height of the device's screen
- Maximum width = twofold width of the device's screen
- For devices with runtime version V13 SP1 the maximum height and the maximum width is equal with the height and width of the device's screen.

Program code: Importing a pop-up screen into a folder

Modify the following program code to import a pop-up screen into the pop-up screen system folder or to a user-defined folder:

```
//Imports a pop-up screen to an HMI device
private static void ImportPopupScreenToHMITarget(HmiTarget hmitarget)
{
    FileInfo info = new FileInfo(string.Format(@"D:\Samples\Screens\PopupScreen.xml"));
    hmitarget.ScreenPopupFolder.ScreenPopups.Import(info, ImportOptions.None);
}
```

8.3.9.12 Exporting a slide-in screen

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The following data and values of the slide-in screen is exported:

Screen tem-plates	Data	
Attributes	Activate	false
	ActiveLayer	0
	BackColor	(182; 182; 182)
	GridColor	(0; 0; 0)
	Dimension	427 The attribute "Dimension" specifies either the width or height of the slide-in screen, depending on the used slide-in screen type.
	LineColor1	(223; 223; 223)
	LineColor2	(32; 32; 32)
	OperatableAreaColor	(128; 128; 128)
	SlideinType	Top, Bottom, Left, Right Slide-in screens do not have a name but a SlideinType.
	Visibility	FadeOut
Compositions	Layers	

Note

Slide-in screens do not have a name but a SlideinType.

The following data is exported for each layer:

Layer	Data	
Attributes	Name,	
	Index	
	VisibleES	
Compositions	ScreenItems	All exportable screen objects are exported.

Program code: Exporting a slide-in screen

Modify the following program code to export an individual slide-in screen from the system folder:

```
//Exports a single slide-in screen
private static void ExportSingleSlideinScreen(HmiTarget hmitarget)
{
    ScreenSlideinSystemFolder systemFolder = hmitarget.ScreenSlideinFolder;
    var screens = systemFolder.ScreenSlideins;
    ScreenSlidein slidein = screens.Find(SlideinType.Bottom);
    if (slidein == null) return;

    FileInfo info = new FileInfo(string.Format(@"D:\Samples\Screens\{0}\{1}.xml"));
    slidein.Export(info, ExportOptions.WithDefaults);
}
```

8.3.9.13 Importing a slide-in screen**Requirement**

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The following data and values of a slide-in screen is imported:

Screen templates	Data
Attributes	Activate = false ActiveLayer = 0 Authorization BackColor = (182; 182; 182) Dimension = 427 The attribute "Dimension" specifies either the width or height of the slide-in screen, depending on which of the two attributes is modifiable for the specified slide-in type. GridColor = (0; 0; 0) LineColor1 = (223; 223; 223) LineColor2 = (32; 32; 32) OperateableAreaColor = (128; 128; 128) SlideinType = Top, Bottom, Left, Right Visibility = FadeOut
Compositions	Layers

The existence of the following attribute is mandatory for the import:

- SlideinType

The following data is imported for each layer:

Layer	Data
Attributes	Name, Index, VisibleES
Compositions	ScreenItems All importable screen objects are imported.

Restrictions

- If a device does not support slide-in screens, the import is cancelled and an Exception is thrown.
 - If a slide-in screen is referenced from another element, the slide-in screen must be referenced via openlink and not via SlideinType, e. g. in system function "ShowSlideinScreen").
- The following table shows the mapping of the attribute "SlideinType" with the corresponding openlink:

SlideinType	Openlink Name
Top	GraphX_Slidein_Top
Right	GraphX_Slidein_Right
Bottom	GraphX_Slidein_Bottom
Left	GraphX_Slidein_Left

Program code: Importing a slide-in screen into a folder

Modify the following program code to import a slide-in screen to the slide-in screen system folder:

```
//Imports a slide-in screen to an HMI device
private static void ImportSlideinScreenToHMITarget(HmiTarget hmitarget)
{
    FileInfo info = new FileInfo(@"D:\Samples\Screens\SlideInScreen.xml");
    hmitarget.ScreenSlideinFolder.ScreenSlideins.Import(info, ImportOptions.None);
}
```

8.3.9.14 Exporting a screen with a faceplate instance

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The following data of a faceplate instance in a screen is exported:

Screen	Data
Attributes	Left, Top, Width, Height, ObjectName, Resizing, TabIndex, FaceplateTypeName
Interface Attributes	All configured interface attributes of a faceplate instance are exported for exportable screen items.
Compositions	<ul style="list-style-type: none">• Animations All movement animations are exported. Tag animations rely on interface attributes.• Events All configured events are exported.

Regard the following specifications for exported attributes of faceplate instance:

- Resizing
The attribute "Resizing" is exported in any case, independent of the export options.
- FaceplateTypeName
The attribute "FaceplateTypeName" identifies the corresponding faceplate type and version, e. g. "Faceplate_1 V 0.0.2".

Note

Faceplate type in a library folder

If a faceplate type is located within a library folder, the complete path and name is required to identify the faceplate type. The keyword "@\$\$@" is used to separate folders and/or faceplate type name, e. g. "Folder_1@\$\$@SubFolder_1@\$\$@Faceplate_1 V 0.0.2".

The following data of inner screen items of a faceplate instance is excluded from the export:

Screen item	Attribute
IO-Field	Flashing on limit violation
Graphic IO-Field	Fit embedded graphic object to screen size

Program code

Modify the following program code to export an individual screen including a faceplate instance:

```
//Exports a single screen including a faceplate instance
private static void ExportSingleScreenWithFaceplateInstance(HmiTarget hmitarget)
{
    ScreenFolder folder = hmitarget.ScreenFolder.Folders.Find("MyScreenFolder");
    ScreenComposition screens = folder.Screens;
    Screen screen = screens.Find("ScreenWithFaceplateName");
    if (screen == null) return;
    {
        FileInfo info = new FileInfo(string.Format(@"D:\Samples\Faceplates\{0}\{1}.xml",
folder.Name, screen.Name));
        screen.Export(info, ExportOptions.WithDefaults);
    }
}
```

8.3.9.15 Importing a screen with a faceplate instance

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

The following data of a faceplate instance in a screen is imported:

Screen	Data
Attributes	Left, Top, Width, Height, ObjectName, Resizing,TabIndex, FaceplateTypeName
Interface Attributes	All configured interface attributes of a faceplate instance are imported for importable screen items.
Compositions	<ul style="list-style-type: none"> • Animations All movement animations are imported. Tag animations rely on interface attributes. • Events All configured events are imported.

The existence of the following attributes is mandatory for the import:

- ObjectName
- FaceplateTypeName

8.3 Importing/exporting data of an HMI device

The following data of inner screen items of a faceplate instance is excluded from the export and import:

Screen item	Attribute
IO-Field	Flashing on limit violation
Graphic IO-Field	Fit embedded graphic object to screen size

Restrictions

- Unknown Faceplate, event or interface attribute
If a faceplate type name, an event name or an interface attribute name is specified in the import file which does not exist in the project, the import is aborted with an Exception.
- Resizing behavior of a faceplate instance
The attribute "Resizing" is imported in any case, independent of the export options.
Examples:
If "Resizing" is set to "KeepRatio", the "Height" attribute is used to calculate the "Width" attribute value.
 - The size of a faceplate type is 100 x 100 pixel. If a faceplate instance is imported with size 300 x 100 pixel and value "FixedSize" is set for the "Resizing" attribute, the import succeeds and the faceplate size is set to 100 x 100 pixel.
 - The size of a faceplate type is 100 x 50 pixel. A faceplate instance is imported with size 100 x 100 pixel and value "KeepRatio" is set for the "Resizing" attribute. The import succeeds and the faceplate size is set to 200 x 100 pixel.

Note

Sizing behavior of imported faceplate instances

The values of "Resizing" and values of interface attributes can affect the size of the imported faceplate instance and even the size of the inclosed screen items.

To avoid unrequested changes of the appearance of a faceplate instance, import a faceplate with the initial size or even without "Width" and "Height" attribute values.

- Deviant interface attribute values
 - If you modify attributes for the import, the last applied interface attribute value is imported.
 - If attributes depend on each other, other attribute values can be changed during the import.
Example: A faceplate includes an I/O field. The attribute "Mode" is connected to an interface attribute. If you first set the mode to "Output" and then set the attribute "Hidden input" to true, the value of "Hidden input" is not applied after import. The first modification set the attribute "Hidden input" to read-only and therefore the value cannot be applied.
 - If a attribute value does not fulfill the restrictions of WinCC, the faceplates type value is displayed.
Example: The display range of a gauge is set from 10 - 80. The attributes "MaximumValue" und "MinimumValue" are configured as interface attributes. If you set a minimum value that exceeds the maximum value, e. g. 100, the faceplate type's value for "MinimumValue" is displayed after import.
 - If an interface attribute is connected with several screen item attributes within the faceplate type, the interface attribute value at the faceplate instance will display the applied attribute value of the first connected screen item.
Example: A faceplate includes two gauge objects with deviant maximum values. The minimum values of both gauges are connected to one single interface attribute. If you first set a minimum value that is applicable for both gauges, both values are set. If you set than a value that is only applicable for the second gauge, the value is only set for the second gauge, but the value of the first gauge is displayed as interface attribute.

Program code: Importing screens including a faceplate instance

Modify the following program code to import a screen including a faceplate instance:

```
//Imports single screen including a faceplate instance
private static void ImportSingleScreenWithFaceplateInstance(HmiTarget hmitarget)
{
    FileInfo info = new FileInfo(@"D:\Samples\Screens\ScreenFaceplate.xml");
    hmitarget.ScreenFolder.Screens.Import(info, ImportOptions.None);
}
```

8.4 Export/Import Watch & Force Table

Requirement:

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- You have opened a project with your TIA Portal Openness application.
See Opening a project (Page 106)

Application

You can use the TIA Portal Openness to export Watch Table and Force Table from TIA Portal to SIMATIC ML and import the Watch Table and Force Table from SIMATIC ML.

Program code: Export Watch Table and Force Table

Modify the following program code to export Watch Table:

```
SoftwareContainer softwareContainer =
((IEngineeringServiceProvider)item).GetService<SoftwareContainer>();
PlcSoftware plcSoftware = softwareContainer.Software as PlcSoftware;
PlcWatchTableComposition exportWatchTables =
plcSoftware.PlcWatchAndForceTableGroup.WatchTables;
PlcWatchTable watchTable = exportWatchTables.Find(watchTableName);
if(watchTable != null)
{
    watchTable.Export((FileInfo) fileInfo, ExportOptions.None);
}
```

Note

Export options in Watch Table should be set (None, WithDefaults, WithReadOnly, WithDefaultsAndReadOnly.). A WatchTable has only one published property i.e. the name. The name is published for read.

Modify the following program code to export Force Table:

```
SoftwareContainer softwareContainer =
((IEngineeringServiceProvider)item).GetService<SoftwareContainer>();
PlcSoftware plcSoftware = softwareContainer.Softw are as PlcSoftware;
PlcForceTableComposition exportForceTables =
plcSoftware.PlcWatchAndForceTableGroup.ForceTables;
PlcForceTable forceTable = exportForceTables[0];
forceTable.Export((FileInfo) fileInfo, ExportOptions.None);
```

Note

There is only one ForceTable in every situation, and it has a read-only name.

Program code: Import Watch Table and Force Table

Modify the following program code to import Watch Table:

```
SoftwareContainer softwareContainer =
((IEngineeringServiceProvider)item).GetService<SoftwareContainer>();
PlcSoftware plcSoftware = softwareContainer.Softw are as PlcSoftware;
PlcWatchTableComposition importWatchTables =
plcSoftware.PlcWatchAndForceTableGroup.WatchTables;
IList<PlcWatchTable> WatchTables = importWatchTables.Import((FileInfo) fileInfo,
ImportOptions.None);
```

Note

The imported WatchTable will be added to the list of WatchTables. Import options in Watch Table should be set to the required one (None, Override).

ForceTables can be imported in a similar way, but it is only allowed to have one ForceTable. If you use the None option instead of Override and a (non-empty) ForceTable already exists, then the import will fail with the following RecoverableException: Only one ForceTable is allowed to be imported. Please use Override importOption to overwrite the existing one.

See also

Connecting to the TIA Portal (Page 76)

Opening a project (Page 106)

8.5 Importing/exporting data of a PLC device

8.5.1 Blocks

8.5.1.1 XML structure of the block interface section

Basic principle

The data in the export file from the import/export is structured with reference to a basic structure. Every import file has to fulfill the basic structural conditions.

The export file includes all edited tags and constants of the interface section of an exported block. All attributes with "ReadOnly=""TRUE"" and "Informative=""TRUE"" are excluded.

If the information is redundant, it must exactly be identical in the import XML file and the project data. Otherwise the import will throw a recoverable exception.

The project data can contain more data than the import XML file, e. g. an external type may have additional members

Only writeable values can be imported via TIA Portal Openness XML.

Depending on the TIA Portal Openness export settings, the export file includes a defined set of attributes and elements. The XML exported from higher versions of the product is not compatible during the import operation in lower version of the TIA Portal.

Basic structure

The interface section of an exported block is covered in the `<Interface>` element in the SimaticML of a block. The root object is the `<Sections>` element, which represents the interface section of an exported block. The sequence of the following description of elements represents the required sequence in the input file.

```

<Interface>
  <Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v1">
    <Section Name="Input">
      <Member Name="input1" Datatype="Bool" Remanence="Volatile" Accessibility="Public">
        <AttributeList>
          ...
        </AttributeList>
      </Member>
      <Member Name="input2" Datatype="Bool" Remanence="Volatile" Accessibility="Public">
        <AttributeList>
          ...
        </AttributeList>
      </Member>
    </Section>
    <Section Name="Output">
      <Member Name="output1" Datatype="Bool" Remanence="Volatile" Accessibility="Public">
        <AttributeList>
          ...
        </AttributeList>
      </Member>
    </Section>
    <Section Name="InOut" />
    <Section Name="Static" />
    <Section Name="Temp" />
    <Section Name="Constant" />
  </Sections>
</Interface>

```

- Section
Section represents a single parameter or local data of a program block
- Member
Member represents the tags or constants used in the program block. Depending of the datatype of a tag, members can be nested or have further structural sub elements.
In case of the data type "ARRAY" the structural element "Subelement Path" represents e. g. the index of the components of an array element.
Only those members are exported, which were edited by the user.
- AttributeList
The `<AttributeList>` includes all defined attributes of a member. Attributes, that are system defined or assigned by a standard value are not listed in the XML structure.
The member attributes `<ReadOnly>` and `<Informative>` are only written to the XML export file if their value is TRUE.

- StartValue

The element <StartValue> is only written, if the default value of the tag or constant is set by the user.

```
...
<Member Name="Static_1" Datatype="Int">
  ...
    <StartValue>1</StartValue>
  </Member>
  ...
```

- Comment

The element <Comment> is written, if it is set by the user. Comments of a tag or constant are exported as multilingual text:

```
...
<Member Name="Static_3" Datatype="Struct">
  <AttributeList>
    <BooleanAttribute Name="ExternalAccessible" SystemDefined="true">false</BooleanAttribute>
    <BooleanAttribute Name="ExternalVisible" SystemDefined="true">false</BooleanAttribute>
    <BooleanAttribute Name="ExternalWritable" SystemDefined="true">false</BooleanAttribute>
  </AttributeList>
  <Comment>
    <MultiLanguageText Lang="de-DE">An individual comment</MultiLanguageText>
  </Comment>
</Member>
...  
...
```

Attributes

- Main attributes

The main attributes are written in the `<Member>` element in the XML structure.

```
...
<Member Name="Static_1" Datatype="User_data_type_1" Remanence="Retain">
| ...
</Member>
...
```

The following table shows the main attributes of a tag or constant at the block interface section.

Name	Datatype	Default	Import condition	Comment
Name	STRING	-	Required	
Datatype	ENUM	-	Required	
Version	STRING	-	Optional	
Remanence	ENUM	NonRetain	-	only written if not default
Accessibility	ENUM	Public	-	pre-defined by the system cannot be changed by the user
Informative	BOOL	FALSE	-	

Members with the flag "Informative" are ignored during import. If the attribute is deleted or set to FALSE, an exception is thrown.

Note

Remanence settings "Set in IDB"

If the remanence value of a tag or constant is "Set in IDB", the remanence set in the IDB has to be the same for all other tags and constants with the remanence value "SetInIDB".

The first imported member with "Set in IDB" attribute defines the expected remanence in the IDB for the following tags and constants with the remanence value "SetInIDB".

- System defined member attributes

Systemdefined member attributes are listed in the element `<AttributeList>`.

Systemdefined member attributes flagged with the `<Informative>` and are ignored during import.

```
...
<Member Name="Static_3" Datatype="Struct">
| <AttributeList>
| | <BooleanAttribute Name="ExternalAccessible" SystemDefined="true">false</BooleanAttribute>
| | <BooleanAttribute Name="ExternalVisible" SystemDefined="true">false</BooleanAttribute>
| | <BooleanAttribute Name="ExternalWritable" SystemDefined="true">false</BooleanAttribute>
| </AttributeList>
| <Comment>
| | <MultiLanguageText Lang="de-DE">An individual comment</MultiLanguageText>
| </Comment>
</Member>
...
```

Name	Type	Default	SimaticML ReadOnly (informative)	Comment
At	string	""	FALSE	Member shares offset with another member in this structure
SetPoint	bool	FALSE	FALSE	Member can be synchronized with workmemory
UserReadOnly	bool	FALSE	TRUE	User cannot change any member attribute (incl. name)
UserDeletable	bool	TRUE	TRUE	Editor does not allow to delete the member
HmiAccessible	bool	TRUE	FALSE	No HMI access, no structure item
HmiVisible	bool	TRUE	FALSE	Filter to reduce the number of members shown in the first place
Offset	int	-	TRUE	DB, FB, FC (Temp). For classic PLCs and for Plus PLCs where the remanence is set to classic.
PaddedSize	int	-	TRUE	DB, FB, FC (Temp). For classic PLCs and for Plus PLCs where the remanence is set to classic. Only for arrays.
HiddenAssignment	bool	FALSE	FALSE	Hide assignment at call if matches with Pre-definedAssignment
PredefinedAssignment	string	""	FALSE	Input for the parameter used when call is placed
ReadOnlyAssignment	bool	FALSE	FALSE	The user cannot change the predefined assignment at the call
UserVisible	bool	TRUE	TRUE	This member is not shown on the UI

Name	Type	Default	SimaticML Re- adOnly (informa- tive)	Comment
HmiReadOnly	bool	TRUE	TRUE	This member is read only for HMI
CodeReadOnly	bool	FALSE	TRUE	-

- User defined attributes

User defined attributes are flagged with <ReadOnly>. Members with this flag are ignored during import. If the flag is deleted or set to FALSE, an exception is thrown.
Unedited user defined attributes are excluded from the export.

Name	Type	Default	SimaticML Re- adOnly (informa- tive)	Comment
CFC	IBlockAttribute	---	FALSE	this is a Payload

Datatype "STRUCT"

The components of the datatype "STRUCT" are represented in the XML structure of an import/export file as nested members:

```

<Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
  <Section Name="Static">
    <Member Name="Static_1" Datatype="Struct">
      <!-- Basic struct -->
      <Member Name="Static_1" Datatype="Int">
        <!-- First Member of struct -->
        <StartValue>1</StartValue>
      </Member>
      <Member Name="Static_2" Datatype="Int">
        <!-- Second Member of struct -->
      </Member>
      <Member Name="Static_3" Datatype="Struct">
        <!-- A subsequent struct -->
        <Member Name="Static_1" Datatype="Int">
          <!-- First Member of the subsequent struct -->
          <StartValue>3</StartValue>
        </Member>
        <Member Name="Static_2" Datatype="Int">
          <!-- Second Member of the subsequent struct -->
        </Member>
      </Member>
    </Section>
  </Sections>

```

Datatype "ARRAY" basic type

The components of the basic datatype "ARRAY" are represented in the XML structure of an import/export file as subelements with the attribute "Path":

```
<Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
  <Section Name="Static">
    <Member Name="Static_1" Datatype="Array[0..2] of Int" Remanence="Retain">
      <!-- Basic Array -->
      <Subelement Path="0">
        <!-- First Array Component-->
        <StartValue>1</StartValue>
      </Subelement>
      <Subelement Path="1">
        <!-- Second Array Component-->
        <StartValue>2</StartValue>
      </Subelement>
    </Member>
  </Section>
</Sections>
```

Datatype "ARRAY" of UDT

The components of the datatype "ARRAY" of an UDT are represented in the XML structure of an import/export file as new `<sections>` element in a `<member>` element. The members in the new section for UDT are within an ARRAY are assigned as subelements with "Path" attribute:

```
<Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
  <Section Name="Static">
    <Member Name="Static_1" Datatype="User_data_type_1" Remanence="Retain">
      <Sections> <!-- Sections including the UDT "User_data_type_1" -->
        <Section Name="None">
          <Member Name="Element_2" Datatype="Int">
            <StartValue>47</StartValue>
          </Member>
        </Section>
      </Sections>
    </Member>
    <Member Name="Static_2" Datatype="Array[0..1] of User_data_type_1">
      <Sections> <!-- Sections including the UDT "User_data_type_1" -->
        <Section Name="None">
          <Member Name="Element_2" Datatype="Int">
            <Subelement Path="0"> <!-- Component of the array -->
              <StartValue>123</StartValue>
            </Subelement>
          </Member>
        </Section>
      </Sections>
    </Member>
  </Section>
</Sections>
```

Datatype "ARRAY" in "ARRAY"

The components of the datatype "ARRAY" in another ARRAY are represented in the XML structure of an import/export file as subelements with the attribute "Path".

The members within another ARRAY are assigned as subelements with "Path" attribute, if the component is edited by the user:

```
<Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
  <Section Name="Static">
    <Member Name="Static_1" Datatype="Array[0..2] of Struct">
      <Member Name="Static_1" Datatype="Int" />
      <Member Name="Static_2" Datatype="Array[0..1, 0..3, -9..-2] of Struct">
        <Member Name="Static_1" Datatype="Int">
          <Subelement Path="0,0,3,-5">
            <StartValue>1</StartValue>
          </Subelement>
        </Member>
        <Subelement Path="0,0,2,-6">
          <Comment>
            <MultiLanguageText Lang="de-DE">A individual comment</MultiLanguageText>
          </Comment>
        </Subelement>
      </Member>
    </Member>
  </Section>
</Sections>
```

PLC data types (UDT)

The XML structure of a PLC data type depends on the TIA Portal Openness export settings.

- ExportOptions.None

Members of PLC data type are only written if the default value of at least one of the components is set by the user. For these members, only the two additional attributes "Name" and "Datatype" are written, to identify the member to which the <StartValue> belongs. Other members and attributes are not written.

- ExportOptions.WithDefaults

The following attributes are always written:

- Name
- Datatype
- ExternalAccessible
- ExternalVisible
- ExternalWritable
- SetPoint
- StartValue

Only written to the XML if it the default value in this type is set by the user. If it only has been set in the PLC data type, it is not written.

- ExportOptions.ReadOnly

For PLC data types this setting will not lead to meaningful result. In combination with other settings it will have no influence on the result.

Overlaid tags

If a tag is overlaid with a new datatype, the members are represented in the XML structure of the new data type. The following XML structure shows a datatype WORD overlaid by an ARRAY of BYTE.

```
<Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
  <Section Name="Input" />
  <Section Name="Output" />
  <Section Name="InOut" />
  <Section Name="Static">
    <Member Name="Static_1" Datatype="Word">
      <!-- Basic Member -->
      <StartValue>16#17</StartValue>
    </Member>
    <Member Name="Static_2" Datatype="Array[0..1] of Byte">
      <AttributeList>
        <StringAttribute Name="At" SystemDefined="true">Static_1</StringAttribute>
      </AttributeList>
      <!-- The AT member -->
      <Subelement Path="0">
        <!-- First overlay byte -->
      </Subelement>
      <Subelement Path="1">
        <!-- Second overlay byte -->
      </Subelement>
    </Member>
  </Section>
  <Section Name="Temp" />
  <Section Name="Constant" />
</Sections>
```

Block Interface

All attributes with `ReadOnly="TRUE"` and `Informative="FALSE"` are excluded. The XML structure of a block interface depends on the TIA Portal Openness export settings.

- `ExportOptions.None`

This setting exports only the modified data or the data that differs from the default.

In case their attribute definition does not specify a default value, the attribute is always written.

The export file also contains all values that are obligatory for the subsequent data import.

- `ExportOptions.WithDefaults`

The following attributes are always written

- `Name`
- `Datatype`
- `HmiAccessible` **exported as ExternalAccessible**
- `HmiVisible` **exported as ExternalVisible**
- `ExternalWritable`
- `SetPoint` **(if applicable)**
- `Offset` **(if applicable)**
- `PaddedSize` **(if applicable)**

All other attributes are only written if their values differ from the default.

The `<StartValue>` element is only written to the XML if it has been explicitly set.

- `ExportOptions.ReadOnly`

For block interfaces this setting will not lead to meaningful result. In combination with other settings it will have no influence on the result.

8.5.1.2 Changes of the object model and XML file format

Introduction

To import a custom created or an edited XML file successfully to the TIA Portal via TIA Portal Openness, the file must correspond to defined schemas.

The XML files always consist of two major parts:

- Interface
- Compile unit

The schemas they have to correspond to are explained in the following.

Interface

An interface can contain multiple sections (e. g. Input, InOut, Static): You can find all of these sections in the schema in the following directory:

- C:\Program Files\Siemens\Automation\Portal V*\PublicAPI\V*\Schemas\SW.InterfaceSections_v3.xsd
- C:\Program Files\Siemens\Automation\Portal V*\PublicAPI\V*\Schemas\SW.Interface.Snapshot .xsd

Compile unit

There are separate schemas for the compile units of GRAPH, LAD/FBD, STL and SCL blocks. You can find these schemas in the following directories:

- GRAPH: C:\Program Files\Siemens\Automation\Portal V*\PublicAPI\V*\Schemas\SW.PlcBlocks.Graph_v4.xsd
- LAD/FBD: C:\Program Files\Siemens\Automation\Portal V*\PublicAPI\V*\Schemas\SW.PlcBlocks.LADFBDB_v3.xsd
- STL: C:\Program Files\Siemens\Automation\Portal V*\PublicAPI\V*\Schemas\SW.PlcBlocks.STL_v3.xsd
- SCL: C:\Program Files\Siemens\Automation\Portal V*\PublicAPI\V*\Schemas\SW.PlcBlocks.SCL_v2.xsd

Subschemas

There are the following additional schema definitions used by all compile units:

- Access
- Common

Access

The Access node describes for example:

- local/global members and constant usages
- FB, FC, Instruction calls
- DBs for calls

You can find the access schema in the following directory:

C:\Program Files\Siemens\Automation\Portal V*\PublicAPI\V*\Schemas\SW.PlcBlocks.Access_v3.xsd

Common

Common contains the commonly used attributes and elements, for example different types of comments, texts and tokens.

You can find the common schema in the following directory:

8.5 Importing/exporting data of a PLC device

C:\Program Files\Siemens\Automation\Portal V*\PublicAPI\V*.\\Schemas
\\SW.Common_v2.xsd

Note

V* refers to the installed version of TIA Portal.

8.5.1.3 Exporting DBs with snapshots

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

You can use the TIA Portal Openness to export the DB's with snapshot values as XML and able to compare the values with different snapshot times. With the compare result, you can manually adapt single start values within the UI and store for a potential recovery. The schema "SW.Interface.Snapshot.xsd" will support you to interpret or process the exported XML-file.

Program code

Modify the following program code to export Snapshot Values by using the Snapshot Service:

```
InterfaceSnapshot interfaceSnapshot = dataBlock.GetService<InterfaceSnapshot>();  
interfaceSnapshot.Export(new FileInfo("C:\\temp\\MyInterfaceSnapshot.xml"),  
ExportOptions.None);
```

The Snapshot Service "InterfaceSnapshot" will be provided in the namespace "Siemens.Engineering.SW.Blocks". The file handling (e.g. if the export directory does not exist; creating of the export directory; if the export directory is readonly; if the export file already exists) will be the same as the standard interface openness export. The Snapshot Service will be supported for Global DBs, Instance DBs and Array DBs.

Note

Export of snapshot values using the Snapshot Service is independent of the standard interface openness export and therefore will not influence the already existing export of the interface members. It will not be possible to import the exported XML

The snapshot values will be exported as following:

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
  <Engineering version="V15 SP1" />
  <DocumentInfo>
    ...
  </DocumentInfo>
  <SW.Blocks.InterfaceSnapshot ID="0">
    <AttributeList>
      <Name>GlobalDB</Name>
      <Snapshot ReadOnly="true"><SnapshotValues>
        <SnapshotValues>
          <Value Path="Static_1" Type="Bool">TRUE</Value>
          <Value Path="Static_2[0]" Type="Int">1</Value>
          <Value Path="Static_2[1]" Type="Int">2</Value>
          <Value Path="Static_2[2]" Type="Int">3</Value>
          <Value Path="Static_3" Type="DTL">DTL#1973-01-01-00:00:00</Value>
          <Value Path="Static_4.Element_1" Type="Int">7</Value>
          <Value Path="Static_4.Element_2[0]" Type="Bool">FALSE</Value>
          <Value Path="Static_4.Element_2[1]" Type="Bool">TRUE</Value>
          <Value Path="Static_4.Element_2[2]" Type="Bool">TRUE</Value>
          <Value Path="Static_4.Element_3.Element_1" Type="Int">5</Value>
          <Value Path="Static_4.Element_3.Element_2.Element_1" Type="Bool">TRUE</Value>
          <Value Path="Static_4.Element_3.Element_2.Element_2[0]" Type="Int">100</Value>
          <Value Path="Static_4.Element_3.Element_2.Element_2[1]" Type="Int">200</Value>
        </SnapshotValues></Snapshot>
        <SnapshotDate ReadOnly="true">2017-12-06T08:04:11.4590585Z</SnapshotDate>
        <StructureModified ReadOnly="true">2017-12-06T08:22:13.3292585Z</StructureModified>
      </AttributeList>
    </SW.Blocks.InterfaceSnapshot>
  </Document>
```

If a DB does not contain any snapshot values, the content of the exported file would look like as following:

```
<SnapshotValues xmlns="http://www.siemens.com/automation/Openness/SW/Interface/Snapshot/v1"></SnapshotValues>
```

See also

[Opening a project \(Page 106\)](#)

8.5.1.4 Exporting blocks with know-how protection

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- PLC is not online.

Application

The resulting XML file is similar to the export file of a block without know-how protection. With PlcBlockProtectionProvider, you can unlock a know-how protected block by providing the password using the Openness API and then export a KHP block with in complete with its code. A block can be imported and then be protected with a password using the API again.

If the block is exported without unlocking, only the public block interface will be exported.

- TIA Portal block -> Unlock -> Export file without protection
- Import file without protection -> Lock -> TIA Portal block

The attribute list of the block indicates that the relevant block is know-how protected.

Program code

Modify the following program code to export the visible data of a block with know-how protection to an XML file:

```
private static void ExportBlock(PlcSoftware plcSoftware)
{
    PlcBlock plcBlock = plcSoftware.BlockGroup.Blocks.Find("MyBlock");
    plcBlock.Export(new FileInfo(string.Format(@"D:\Samples\{0}.xml", plcBlock.Name)),
    ExportOptions.WithDefaults);
}
```

8.5.1.5 Export/Import of SCL blocks

SCL statements with export XML tags

The export operation of SCL blocks exports its equivalent XML tags based on the type of SCL statements. This operation supports the SCL networks of SCL statements in LAD/FBD blocks of SCL statements. The SCL statements are classified as text elements, operands, expressions, control, etc. The schema SW.PlcBlocks.SCL_v2.xsd is available to support you to process the SCL block with export XML tags. The SCL block statements with their corresponding exported XML tags and attributes are given below.

New line

New lines in SCL blocks are represented as NewLine XML tag.

- Contains unsigned Num attribute with default value 1.
- Num attribute does not have value 0.
- Supported only for SCL.

SCL block	XML tag
	<NewLine Num="2" />

Blank

Blank spaces in SCL blocks are represented as Blank XML tag.

- Contains unsigned Num attribute with default value 1.
- Num attribute does not have value 0.
- Supported only for SCL.
- Does not support Integer attribute available in other languages of STEP 7.

SCL block	XML tag
	<Blank Num="2" />

Indentation of SCL block statements

In TIA Portal settings, you can modify the indentation of SCL code by accessing Options/Settings/General/Script/text editors. The following table defines the type of indentation based on the ident mode.

Ident mode	Result
None	Import operation adds the spaces as available in the source files.
Paragraph or Smart	Import operation adds the specified ident spaces in the imported file.

Based on the chosen indentation, the imported SCL block XML file are indented.

Comment

Single-line and multi-line comments in SCL blocks are represented as LineComment XML tag.

- Only LineComment tag (for single language comment) is used in SCL.
- Comment tag (for multiple language comment) is not used in SCL.
- Contains Inserted attribute with default value false
- Inserted="false" indicates "//" single comment in SCL block.
- Inserted="true" indicates "(*)" multi-line comment in SCL block.

- NoClosingBracket="true" indicates comment without closing braces in SCL block. This attribute is optional and has default value as false.
- XML does not indicate comment hierarchy in SCL block.

SCL block	XML tag
// one line comment	<LineComment> <Text>one line comment</Text> </LineComment>
(* one line comment second line *)	<LineComment Inserted="true"> <Text>one linecomment secondline</Text> </LineComment>
(* first comment (* second comment *) end first comment *)	<LineComment Inserted="true"> <Text> first comment (* second comment *) end first comment</Text> </LineComment > The nested comment is part of outer comment text.
(* comment without closing bracket	<LineComment Inserted="true" NoClosingBracket="true"> <Text> comment without closing bracket</Text> </LineComment >

Region

Regions in SCL blocks are represented as Token XML tag.

- Text XML tag represents the region_name.
- The Text attribute of the Token XML tag is case in-sensitive.
- The import operation is case in-sensitive, and the editor displays the keywords as configured in the TIA Portal settings.
- If the end_region keyword ends with ";" (semi-colon) in SCL block, the symbol ";" is placed in Text XML tag.

SCL block	XML tag
<pre>region myregion ... end_region here is the end of myregion</pre>	<Token Text="REGION" /> <Blank /> <Text>myregion</Text> <NewLine /> ... <Token Text="END_REGION" /> <Blank /> <Text>here is the end of myregion</Text> <NewLine />
<pre>region // here are no blanks ... end_region</pre>	<Token Text="REGION" /> <NewLine /> <LineComment .../> <Token Text="END_REGION" /> <NewLine />
<pre>region ... end_region;</pre>	<Token Text="REGION" /> <NewLine /> ... <Token Text="END_REGION" /> <Text>;</Text> <NewLine />

Pragma

Pragma in SCL blocks are represented as Token XML tag. The parameters are represented in Access XML tag with Scope attribute as LiteralConstant.

SCL block	XML tag
<pre>{PRAGMA_BEGIN 'Param1', 'Param2' (*parm 2*) // something else {PRAGMA_END}</pre>	<pre><Token Text="{" /> <Token Text="PRAGMA_BEGIN" /> <Blank /> <Access Scope="LiteralConstant"> <Constant> <ConstantValue>'Param1'</ ConstantValue> </Constant> </Access> <Token Text="," /> <Blank /> <Access Scope="LiteralConstant"> <Constant> <ConstantValue>'Param2'</ ConstantValue> </Constant> </Access> <Blank /> <LineComment Inserted="True"> <Text>param 2</Text> </LineComment> <Token Text="," /> <Blank /> <Token Text="}" /> <NewLine /> <LineComment> <Text> something else</Text> </LineComment> <NewLine /> <Token Text="{" /> <Token Text="PRAGMA_END" /> <Token Text="}" /></pre>

Constants: Literal constants

The constants in SCL blocks are represented by Access XML tag.

- The Scope attribute can have values like LiteralConstant, TypedConstant, LocalConstant, and GlobalConstant.
- The name of constants preceded by "#" are ignored in XML.
- The "#" is added during the import operation of XML.
- The value of global constants represented by quotes are ignored in XML.
- The quotes are added during the import operation of XML.

Type of constant	SCL block	XML tag
Literal constant: Integer	#Out := 10;	<pre><Access Scope="LiteralConstant"> <Constant> <ConstantValue>10</ConstantValue> <ConstantTypeInformative="true">LINT</ ConstantType> </Constant> </Access></pre>
Literal constant: String	#myString := 'Hello world';	<pre><Access Scope="LiteralConstant"> <Constant> <ConstantValue>Hello world</ ConstantValue> <ConstantTypeInformative="true">STRING</ ConstantType> </Constant> </Access></pre>
Literal constant: Typed	#Out := int#10;	<pre><Access Scope="TypedConstant"> <Constant> <ConstantValue>int#10</ConstantValue> </Constant> </Access></pre> <p>Format of XML exported in ExportOptions.ReadOnly setting.</p> <pre><Access Scope="TypedConstant"> <Constant> <ConstantValue>int#10</ConstantValue> <StringAttribute Name="Format" Informative="true">Dec_signed</ StringAttribute> <StringAttribute Name="FormatFlags" Informative="true">TypeQualifier</ StringAttribute> </Constant> </Access></pre>
Local constant	#Out := #mylocal;	<pre><Access Scope="LocalConstant"> <Constant Name="mylocal" /> </Access></pre> <p>Format of XML exported in ExportOptions.ReadOnly setting.</p> <pre><Access Scope="LocalConstant"> <Constant Name="mylocal"> <ConstantType Informative="true">Int</ ConstantType> <ConstantValue Informative="true">10</ ConstantValue> <StringAttribute Name="Format" Informative="true">Dec_signed</ StringAttribute> </Constant> </Access></pre>

Type of constant	SCL block	XML tag
Global constant	#Out := "myglobal";	<pre><Access Scope="GlobalConstant"> <Constant Name="myglobal" /> </Access></pre> <p>Format of XML exported in ExportOptions.ReadOnly setting.</p> <pre><Access Scope="GlobalConstant"> <Constant Name="myglobal"> <ConstantType Informative="true">Int</ConstantType> <ConstantValue Informative="true">10</ConstantValue> <StringAttribute Name="Format" Informative="true">Dec_signed</StringAttribute> </Constant> </Access></pre>

The address constants are not supported in SCL blocks, and it is ignored in this table.

Variables

The local and global variables in SCL blocks are represented by Access XML tag.

- The Scope attribute has values of LocalVariable and GlobalVariable
- The XML tags for assigning the value 10 is ignored here.

Type of variable	SCL block	XML tag
Local variable	#Out := 10;	<pre><Access Scope="LocalVariable"> <Symbol> <Component Name="Out" /> </Symbol> </Access></pre>
Global variable	"Tag_3":= 10;	<pre><Access Scope="GlobalVariable"> <Symbol> <Component Name="Tag_3" /> </Symbol> </Access></pre> <p>Format of XML exported in ExportOptions.ReadOnly setting.</p> <pre><Access Scope="GlobalVariable"> <Symbol> <Component Name="Tag_3" /> <Address Area="Memory" Type="Int" BitOffset="96" Informative="true" /> </Symbol> </Access></pre>

Expressions

The simple expressions in SCL blocks are represented by Access XML tag. The Scope attribute has value of LocalVariable for the expressions

SCL block	XML tag
#a := #b + #c;	<Access Scope="LocalVariable"> <Symbol> <Component Name="a" /> </Symbol> </Access> <Blank /> <Token text=":=" /> <Blank /> <Access Scope="LocalVariable"> <Symbol> <Component Name="b" /> </Symbol> </Access> <Blank /> <Token text="+" /> <Blank /> <Access Scope="LocalVariable"> <Symbol> <Component Name="c" /> </Symbol> </Access> <Token text=";" />

Control structures in SCL blocks

The control statements like IF, CASE, FOR, WHILE, REPEAT, GOTO, EXIT, CONTINE, and RETURN are represented by Token XML tag.

- The conditional symbols used in SCL block such as >, <, & are represented as escape sequences (< > &&) in XML.
- These combination of XML tags are applicable only for SCL blocks. An exception is thrown for other languages.

Name of the block	SCL block	XML tag
IF	IF #a<#c THEN ; END_IF;	<Token Text="IF" /> <Blank /> <Access Scope="LocalVariable"> <Symbol> <Component Name="a" /> </Symbol> </Access> <Token Text="<" /> <Access Scope="LocalVariable"> <Symbol> <Component Name="c" /> </Symbol> </Access> <Blank /> <Token Text="THEN" /> <NewLine /> <Blank Num="4" /> <Token Text=";" /> <NewLine /> <Token Text="END_IF" /> <Token Text=";" />

Name of the block	SCL block	XML tag
CASE	<pre>CASE #a OF 1 (*test*): // Statement section case 1 ; 2..4: // Statement section case 2 to 4 ; ELSE // Statement section ELSE ; END_CASE;</pre>	<Tok en Text="CASE" /><Blank /> <Access Scope="LocalVariable"> <Symbol> <Component Name="a" /> </Symbol> </Access> <Blank /> <Token Text="OF" /> <NewLine /> <Blank Num="2"/> <Access Scope="LiteralConstant"> <Constant> <ConstantValue>1</ConstantValue> <ConstantType Informative="true">LINT</ConstantType> </Constant> </Access> <Blank /> <LineComment Inserted="true"> <Text>test</Text> </LineComment > <Token Text=":" /> <Blank /> <LineComment> <Text> Statement section case 1</Text> </LineComment > <NewLine /> <Blank Num="4"/> <Token Text=";" /> <NewLine /> <Blank Num="2"/> <Access Scope="LiteralConstant"> <Constant> <ConstantValue>2</ConstantValue> <ConstantType Informative="true">LINT</ConstantType> </Constant> </Access> <Token Text=".." /> <Blank Num="2"/> <Access Scope="LiteralConstant"> <Constant> <ConstantValue>4</ConstantValue> <ConstantType Informative="true">LINT</ConstantType> </Constant> </Access> <Blank /> <LineComment> <Text> Statement section case 2 to 4</Text> </LineComment > <NewLine />

Name of the block	SCL block	XML tag
		<Blank Num="4"/> <Token Text=";" /> <NewLine /> <Blank Num="2"/> <Token Text="ELSE" /> <NewLine /> <Blank Num="4"/> <Token Text=";" /> <NewLine /> <Token Text="END_CASE" /> <Token Text=";" />
FOR	FOR #i := #a TO #b DO // Statement section FOR ; END_FOR;	<Token Text="FOR" /> <Blank /> <Access Scope="LocalVariable"> <Symbol> <Component Name="i" /> </Symbol> </Access> <Blank /> <Token Text=":=" /> <Blank /> <Access Scope="LocalVariable"> <Symbol> <Component Name="a" /> </Symbol> </Access> <Blank /> <Token Text="TO" /> <Blank /> <Access Scope="LocalVariable"> <Symbol> <Component Name="b" /> </Symbol> </Access> <Blank /> <Token Text="DO" /> <NewLine /> <Blank Num="2" /> <LineComment> <Text> Statement section FOR</Text> </LineComment> <NewLine /> <Blank Num="2" /> <Token Text=";" /> <NewLine /> <Token Text="END_FOR" /> <Token Text=";" />

Name of the block	SCL block	XML tag
WHILE	WHILE #a<#b DO // Statement section WHILE ; END WHILE;	<Token Text="WHILE" /> <Blank /> <Access Scope="LocalVariable"> <Symbol> <Component Name="a" /> </Symbol> </Access> <Token Text="<" /> <Access Scope="LocalVariable"> <Symbol> <Component Name="b" /> </Symbol> </Access> <Blank /> <Token Text="DO" /> <NewLine /> <Blank Num="2" /> <LineComment> <Text> Statement section WHILE</Text> </LineComment > <NewLine /> <Blank Num="2" /> <Token Text=";" /> <NewLine /> <Token Text="END WHILE" /> <Token Text=";" />

Name of the block	SCL block	XML tag
REPEAT	<pre>REPEAT // Statement section REPEAT ; UNTIL #a<#b END_REPEAT;</pre>	<pre><Token Text="REPEAT" /> <NewLine /> <Blank Num="2" /> <LineComment> <Text> Statement section REPEAT</Text> </LineComment > <NewLine /> <Blank Num="2" /> <Token Text=";" /> <NewLine /> <Token Text="UNTIL" /> <Blank /> <Access Scope="LocalVariable"> <Symbol> <Component Name="a" /> </Symbol> </Access> <Token Text="<" /> <Access Scope="LocalVariable"> <Symbol> <Component Name="b" /> </Symbol> </Access> <Blank /> <Token Text="END_REPEAT" /> <Token Text=";" /></pre>

Name of the block	SCL block	XML tag
GOTO	here // well : // this is goto statement	<p>XML example for GOTO label definition</p> <pre><Blank Num="3"/> <Access Scope="Label"> <Label Name="here"> <NewLine /> <Blank Num="3"/> <LineComment> <Text> well</Text> </LineComment> <NewLine /> <Token Text=":" /> <Blank /> </Label> </Access> <LineComment> <Text> this is goto statement</Text> </LineComment></pre>
	GOTO (*comment*) here;	<p>XML example for GOTO label usage</p> <pre><Token Text="GOTO" /> <Blank /> <LineComment inserted="true"> <Text>comment</Text> </LineComment> <Blank /> <Access Scope="Label"> <Label Name="here" /> </Access> <Token Text=";" /></pre>

Referencing attributes

The SCL block referencing attributes are represented by AccessModifier attribute of Component tag.

- For simple referencing, AccessModifier has value as Reference.
- For array referencing, AccessModifier has value as ReferenceToArray.

SCL block	XML tag
RefToUDT^ (*RefToUDT*) .element	<pre> <Symbol> <Component Name="RefToUDT" AccessModifier="Reference" /> <Token Text="^" /> <LineComment Inserted="True"> <Text>RefToUDT</Text> </LineComment> <Token Text="." /> <Component Name="element" /> </Symbol></pre>
RefToArrayOfUDT^(*RefToArrayOfUDT*)[#i].element	<pre> <Symbol> <Component Name="RefToArrayOfUDT" AccessModifier="ReferenceToArray" /> <Token Text="^" /> <LineComment Inserted="True"> <Text>RefToArrayOfUDT</Text> </LineComment> <Token Text="[" /> <Access Scope=LocalVariable> <Symbol> <Component Name="i" /> </Symbol> </Access> <Token Text="]" /> </Component> <Token Text="." /> <Component Name="element" /> </Symbol></pre>

8.5.1.6 Export/Import of structured types of SCL blocks

SCL structured types with export XML tags

In the SCL structured types, you can add blanks, new lines, and comments in the SCL statements. The SCL strctured statements with its corresponding exported XML tags and attributes are given below.

Global access

In SCL statements, the global access variables and constants are represented in quotes. The comments written between the variables and address parts are represented by LineComment XML tag.

SCL block	XML tag
"Data_block_1".(*comment 1*)Static_1(*comment 2*).Static_2	<pre><Access Scope="GlobalVariable"> <Symbol> <Component Name="Data_block_1" /> <Token Text="." /> <LineComment Inserted="True"> <Text>comment 1</Text> </LineComment> <Component Name="Static_1" /> <LineComment Inserted="True"> <Text>comment 2</Text> </LineComment> <Token Text="." /> <Component Name="Static_2" /> </Symbol> </Access></pre>
"Data_block_1".Static_1 := 10	<p>Format of XML exported in ExportOptions.None setting.</p> <pre><Access Scope="GlobalVariable"> <Symbol> <Component Name="Data_block_1" /> <Token Text="." /> <Component Name="Static_1" /> </Symbol> </Access></pre> <p>Format of XML exported in ExportOptions.ReadOnly setting.</p> <pre><Access Scope="GlobalVariable"> <Symbol> <Component Name="Data_block_1" /> <Token Text="." /> <Component Name="Static_1" /> <Address Area="DB" Type="Word" BlockNumber="1" BitOffset="0" Informative="true" /> </Symbol> </Access></pre>

Usage of Quotes and

The quotes used in the first level describes the type of variable, and used to escape special characters in SCL statements. When quotes are used in first level, it defines the variable as global variable. If the quotes are used after #, they represent the escape sequence of special characters like #, and spaces.

- To represent the differential usage, XML file uses BooleanAttributes tag with Name attribute. The Name contain values such as HasQuotes and HasHash.
- To define structure in scope attribute, # is defined.
- These values are applicable only for SCL.
- The default values for these tags were FALSE, but the values never gets exported in ExportOptions.WithDefaults settings too.

SCL block	XML tag
"a".#b."c".#"d"	<Access Scope="GlobalVariable"> <Symbol> <Component Name="a" /> <Token Text="." /> <Component Name="b"> <BooleanAttribute Name="HasHash">TRUE</ BooleanAttribute> </Component> <Token Text="." /> <Component Name="c"> <BooleanAttribute Name="HasQuotes">TRUE</ BooleanAttribute> </Component> <Token Text="." /> <Component Name="d"> <BooleanAttribute Name="HasQuotes">TRUE</ BooleanAttribute> <BooleanAttribute Name="HasHash">TRUE</ BooleanAttribute> </Component> </Symbol> <Access />

Array

SCL allows to add comment within the array indexes around "[" and "]". To mark the existence of array, XML file uses AccessModifier attribute in Component tag.

- If Accessmodifier contains the value Array, then a child tag Access is mandatory to indicate the index variable of the array.
- The default value for AccessModifier is None.

SCL block	XML tag
#a.b[#i+#j,#k+#l].c	<Access Scope="LocalVariable"> <Symbol> <Component Name="a" /> <Token Text="." /> <Component Name="b" AccessModifier="Array" /> <Token Text="[" /> <Access Scope=LocalVariable> <Symbol> <Component Name="i" /> </Symbol> </Access> <Token Text="+" /> <Access Scope=LocalVariable> <Symbol> <Component Name="j" /> </Symbol> </Access> <Token Text="," /> <Access Scope=LocalVariable> <Symbol> <Component Name="k" /> </Symbol> </Access> <Token Text="+" /> <Access Scope=LocalVariable> <Symbol> <Component Name="l" /> </Symbol> </Access> <Token Text="]" /> </Component> <Token Text="." /> <Component Name="c" /> </Symbol> </Access>

Absolute Access

SCL allows different types of access such as absolute, absolute offset, mixed (database and member variable), slice, peripheral, and direct type. The absolute access specifiers are represented by Address tag in XML.

- The % character of the DB is not written in XML. It is created automatically during the import.
- Blanks are allowed between the address parts

SCL Block	XML Tag
%DB20 . DBW10	<pre><Access Scope="Address"> <Symbol> <Address Area="DB" BlockNumber="20" /> <Blank /> <Token Text="." /> <Blank /> <Address Area="DB" BitOffset="80" Type="Word"/> </Symbol> </Access></pre>
%DB20.DBX10.3 := true;	<p>The following XML is valid for all languages except SCL.</p> <pre><Access Scope="Address"> <Address Area="DB" BlockNumber="20" BitOffset="83" Type="Bool" /> </Access></pre> <p>The following XML is valid for SCL.</p> <pre><Access Scope="Address"> <Symbol> <Address Area="DB" BlockNumber="20" /> <Token Text="." /> <Address Area="DB" BitOffset="83" Type="Bool"/> </Symbol> </Access></pre>

Absolute offset

In STL, AbsoluteOffset tag represents the absolute offset access. In SCL, Address tag is used for absolute access.

SCL Block	XML Tag
#Input_DB_ANY.%DBX2.3 := TRUE;	<pre><Access Scope="LocalVariable"> <Symbol> <Component Name="Input_DB_ANY" /> <Token Name="." /> <Address BitOffset="19" Type="Bool" /> </Symbol> </Access></pre>

Slicing

In SCL, the SliceAccessModifier attribute is not supported and the slicing is represented by Token tag.

SCL Block	XML Tag
"tag_1"(*1*) . (*2*) member (*3*) . (*4*) %x1	<Access Scope="GlobalVariable"> <Symbol> <Component Name="tag_1" /> <LineComment Inserted="True"> <Text>1</Text> </LineComment> <Token Text="." /> <LineComment Inserted="True"> <Text>2</Text> </LineComment> <Component Name="member"/> <LineComment Inserted="True"> <Text>3</Text> </LineComment> <Token Text="." /> <LineComment Inserted="True"> <Text>4</Text> </LineComment> <Token Text="%x1" /> </Symbol> </Access>

Peripheral access

The peripheral access is represented by Token tag.

SCL Block	XML Tag
"tag_1"(*1*) . (*2*) member:P	<Access Scope="GlobalVariable"> <Symbol> <Component Name="tag_1" /> <LineComment Inserted="True"> <Text>1</Text> </LineComment> <Token Text="." /> <LineComment Inserted="True"> <Text>2</Text> </LineComment> <Component Name="member"/> <Token Text=":P" /> </Symbol> </Access>

Direct type access

The TypeOf and TypeOfDB instructions are handled either with system type or user defined type. The types are represented in Access tag with Scope attribute containing values of SystemType and UserType.

SCL Block	XML Tag
Example for system type if TypeOf(#inVariant) = TO_SpeedAxis then ... end_if	<Token text="=" /> </Blank> <Access Scope="SystemType"> <DataType>TO_SpeedAxis</DataType> </Access>
Example for user defined type if TypeOf(#inVariant) = "aUserDefinedType" then ... end_if	<Token text="=" /> </Blank> <Access Scope="UserType"> <DataType>aUserDefinedType</ DataType> </Access>

8.5.1.7 Export/Import of SCL call blocks

SCL call blocks with export XML tags

SCL call parameters are represented by Parameter tag in XML. The informative attribute is used to represent the non-assigned FB parameters and return values such as timestamp, flag information, etc. The XML format follows the same arbitrary order followed in the SCL block.

An example for block call is given below.

SCL block	XML tag
#Callee_Instance(Input_1 := 5);	Format of XML exported in ExportOptions.None setting <pre> <Access Scope="Call"> <CallInfo BlockType="FB"> <Instance Scope="LocalVariable"> <Component Name="Callee_Instance" /> </Instance> <Token text="("> <Parameter Name="Input_1"> <Blank /> <Token text=":=" /> <Blank /> <Access Scope="LiteralConstant"> <Constant> <ConstantType>Int</ ConstantType> <ConstantValue>5</ ConstantValue> </Constant> </Access> </Parameter> <Token text=")"> </CallInfo> </Access> <Token text=";" /> </pre>
	Format of XML exported in ExportOptions.ReadOnly setting <pre> <Access Scope="Call"> <CallInfo BlockType="FB"> <IntegerAttribute Name="BlockNumber" Informative="true">1</ IntegerAttribute> <DateAttribute Name="ParameterModifiedTS" Informative="true">2016-10-24T08:27: 34</DateAttribute> <Instance Scope="LocalVariable"> <Component Name="Callee_Instance" /> </Instance> <Token text="("> <Parameter Name="Input_1"> <StringAttribute Name="InterfaceFlags" Informative="true">S7_Visible</ StringAttribute> <Blank /> <Token text=":=" /> </Parameter> </Token> </pre>

SCL block	XML tag
	<Blank /> <Access Scope="LiteralConstant"> <Constant> <ConstantType>Int</ConstantType> <ConstantValue>5</ConstantValue> </Constant> </Access> </Parameter> <Token text=")" /> </CallInfo> </Access> <Token text=";" />

Unconnected parameters example

The FB has 4 parameters where a, b, c, and d. b and d are not connected.

SCL block	XML tag
"Block_4_DB" (a:=TRUE,c:=TRUE);	<Access Scope="Call"> <CallInfo Name="Block_4"> BlockType="FB" <Instance Scope="GlobalVariable"> <Component Name="Block_4_DB" /> </Instance> <Token text="(" /> <Parameter Name="a"> <Token text=":=" /> <Access Scope="LiteralConstant"> <Constant> <ConstantType>Bool</ConstantType> <ConstantValue>TRUE</ConstantValue> </Constant> </Access> </Parameter> <Token text="," /> <Parameter Name="b" Informative="true"/> <Parameter Name="c" > <Token text=":=" /> <Access Scope="LiteralConstant"> <Constant> <ConstantType>Bool</ConstantType> <ConstantValue>True</ConstantValue> </Constant> </Access> </Parameter> <Parameter Name="d" Informative="true"/> <Token text=")" /> </CallInfo> </Access>

One parameter example

SCL block allows you to omit the parameter name. This parameter is represented as NamelessParameter tag. The NamelessParameter tag has no attributes and it is applicable only for SCL.

SCL block	XML tag
"Block_4_DB" (TRUE);	<Access Scope="Call"> <CallInfo Name="Block_4" BlockType="FB"> <Instance Scope="GlobalVariable"> <Component Name="Block_4_DB" /> </Instance> <Token text="("> <NamelessParameter> <Access Scope="LiteralConstant"> <Constant <ConstantType>Bool</ConstantType> <ConstantValue>TRUE</ConstantValue> </Constant> </Access> </NamelessParameter> <Token text=")" /> </CallInfo> </Access>

Expression as actual parameter

SCL block	XML tag
#Callee_Instance(Input_1 := #a+3);	<Access Scope="Call"> <CallInfo BlockType="FB"> <Instance Scope="LocalVariable"> <Component Name="Callee_Instance" /> </Instance> <Token text="(" /> <Parameter Name="Input_1"> <Blank /> <Token text=":=" /> <Blank /> <Access Scope="LocalVariable"> <Symbol> <Component Name="a" /> </Symbol > </Access> <Token text="+" /> <Access Scope="LiteralConstant"> <Constant> <ConstantType>Int</ ConstantType> <ConstantValue>3</ ConstantValue> </Constant> </Access> </Parameter> <Token text=")" /> </CallInfo> </Access> <Token text=";" />

Expression as actual parameter without formal parameter

SCL block	XML tag
#Callee_Instance (#a+3);	<Access Scope="Call"> <CallInfo BlockType="FB"> <Instance Scope="LocalVariable"> <Component Name="Callee_Instance" /> </Instance> <Token text="(" /> <NamelessParameter> <Access Scope="LocalVariable"> <Symbol> <Component Name="a" /> </Symbol> </Access> <Token text="+" /> <Access Scope="LiteralConstant"> <Constant> <ConstantType>Int</ ConstantType> <ConstantValue>3</ ConstantValue> </Constant> </Access> </NamelessParameter> <Token text=")" /> </CallInfo> </Access> <Token text=";" />

Function call

SCL block	XML tag
#myInt := "MyFunction"(Param_1 := 1, Param_2 := 15, Param_3 := TRUE);	<Access Scope="LocalVariable"> <Symbol> <Component Name="myInt" /> </Symbol> </Access> <Blank /> <Token text=":=" /> <Blank /> <Access Scope="Call"> <CallInfo Name="MyFunction" BlockType="FC"> <Token text="(" /> <Parameter Name="Param_1"> ... </Parameter> </CallInfo> </Access>

Absolute call

In SCL, the call can be initiated using absolute address of the DB. Due to absolute address, the Name attribute of CallInfo node is empty.

A recoverable exception is thrown by the import of

- An "Address" node is available, with valid value of Name attribute.
- Non-existence of Address node with no valid value of Name attribute.

SCL block	XML tag
%DB20(...);	<pre><Access Scope="Call"> <CallInfo Name="" BlockType="FB"> <Instance Scope="GlobalVariable"> <Address Area="DB" BlockNumber="20" /> </Instance> <Token text="(" /> <Parameter> ... </Parameter> <Token text=")" /> </CallInfo> </Access></pre>

Instruction

The instruction in SCL block is checked in system library during the import operation, and the instruction versions are not exported in export operation.

The general instruction type is given below.

SCL block	XML tag
#myInt := ATTACH(OB_NR := 1, EVENT := 15, ADD := TRUE);	<p>Format of XML exported in ExportOptions.ReadOnly setting</p> <pre> <Access Scope="LocalVariable"> <Symbol> <Component Name="myInt" /> </Symbol> </Access> <Blank /> <Token text=":=" /> <Blank /> <Access Scope="Call"> <Instruction Name="ATTACH"> <Token text="("> <Parameter Name="OB_NR"> <Blank /> <Token text=":=" /> <Blank /> <Access Scope="LiteralConstant"> <Constant> <ConstantType>OB_ATT</ ConstantType> <ConstantValue>1</ ConstantValue> </Constant> </Access> </Parameter> <Token text="," /> <Blank /> <Parameter Name="EVENT"> <Blank /> <Token text=":=" /> <Blank /> <Access Scope="LiteralConstant"> <Constant> <ConstantType>EVENT_ATT</ ConstantType> <ConstantValue>15</ ConstantValue> </Constant> </Access> </Parameter> <Token text="," /> <Blank /> <Parameter Name="ADD"> <Blank /> <Token text=":=" /> <Blank /> <Access Scope="LiteralConstant"> <Constant> <ConstantType>Bool</ ConstantType> <ConstantValue>TRUE</ ConstantValue> </Access> </Parameter> </Instruction> </Call> </pre>

SCL block	XML tag
	</Constant> </Access> </Parameter> <Parameter Name="RET_VAL" Informative="true" /> <Token text=")" /> </Instruction> </Access> <Token text=";" />

Instruction with template

When the template parameter is complements the instruction name, the export of the template parameter is necessary. If a "TemplateValue" tag with attribute Type="Type" follows the Instruction tag, the import operation concatenates the template value to the instruction name.

SCL block	XML tag
"tag_4" := MIN_DINT(IN1:="Tag_1", IN2:="Tag_2", IN3:="Tag_3");	<Access Scope="GlobalVariable"> <Symbol> <Component Name="Tag_4" /> </Symbol> </Access> ... <Access Scope="Call"> <Instruction Name="MIN"> <TemplateValue Name="value_type" Type="Type">DInt</ TemplateValue> ... <Parameter Name="IN1"> ... <Access Scope="GlobalVariable"> <Symbol> <Component Name="Tag_1" /> </Symbol> </Access> </Parameter> ... <Parameter Name="IN2">... <Access Scope="GlobalVariable"> <Symbol> <Component Name="Tag_2" /> </Symbol> </Access> </Parameter> ... <Parameter Name="IN3"> ... <Access Scope="GlobalVariable"> <Symbol> <Component Name="Tag_3" /> </Symbol> </Access> </Parameter> ... </Instruction> </Access> ...

Conversion

For conversion functions, the real instruction name and its template values are not exported. Instead, the name used in the SCL block is exported.

SCL block	XML tag
#output_1 := TIME_TO_S5TIME(#input_1);	<Access Scope="LocalVariable"> <Symbol> <Component Name="output_1" /> </Symbol> </Access> ... <Access Scope="Call"> <Instruction Name="TIME_TO_S5TIME"> <Token text="(" /> <NamelessParameter> <Access Scope="LocalVariable"> <Symbol> <Component Name="input_1" /> </Symbol> </Access> </NamelessParameter> <Token text=")" /> </Instruction> </Access> ...

Instruction with instance

The instance and instruction are separated by blanks. Blanks are optional, and they can be represented by new lines and comments. The instruction TON is represented by Name attribute of Instruction tag.

SCL block	XML tag
IEC_Timer_0_DB . TON (IN:="Tag_1", PT:="Tag_2");	<Access Scope="GlobalAccess"> <Symbol> <Component Name="IEC_Timer_0_DB" /> </Symbol > </Access> <Blank /> <Token text="." /> <Blank /> <Access Scope="Call"> <Instruction Name="TON"> <Blank /> <Token text="(" /> <Parameter Name="IN"> <Access Scope="GlobalVariable"> <Symbol> <Component Name="Tag_1" /> </Symbol> </Access> </Parameter> <...> <Token text=")" /> </Instruction> </Access> <Token text=";" />

Alarm constant

The alarm constants are only used in S7 400 PLCs, and the exported XML is similar to other languages.

SCL block	XML tag
"Block_1_DB"(16#0000_0001);	<p>Format of XML exported in ExportOptions.None setting</p> <pre> <Access Scope="Call"> <CallInfo Name="Block_1" BlockType="FB"> <Instance Scope="GlobalVariable"> <Component Name="Block_1_DB" /> </Instance> <NamelessParameter> <Access Scope="AlarmConstant"> <Constant> <ConstantType>C_Alarm_8</ ConstantType> <ConstantValue>16#0000_0001</ ConstantValue> </Constant> </Access> </NamelessParameter > </CallInfo> </Access></pre>
	<p>Format of XML exported in ExportOptions.ReadOnly setting</p> <pre> <Access Scope="Call"> <CallInfo Name="Block_1" BlockType="FB"> <Instance Scope="GlobalVariable"> <Component Name="Block_1_DB" /> </Instance> <NamelessParameter> <Access Scope="AlarmConstant" > <Constant> <ConstantValue>16#00000001</ ConstantValue> <ConstantType>C_Alarm</ ConstantType> <StringAttribute Name="Format" Informative="true">Hex</ StringAttribute> </Constant> </Access> </NamelessParameter> </CallInfo> </Access></pre>

ENO (Enable Output)

To support the ENO construct in SCL block, an attribute named "Scope" with value "PredefinedVariable" is used in the "Access" tag. It also contains the "PredefinedVariable" tag as child of the Access tag.

- The "PredefinedVariable" tag has one mandatory "Name" attribute.
- The scope "PredefinedVariable" and the tag "PredefinedVariable" are only allowed for SCL.

SCL block	XML tag
Call(..., ENO => ENO);	<pre> <Access Scope="Call"> <CallInfo BlockType="FC"> <Token text="("> ... <Token text="," /> <Blank /> <Parameter Name="ENO"> <Blank /> <Token text="=>" /> <Blank /> <Access Scope="PredefinedVariable"> <PredefinedVariable Name="ENO" /> </Access> </Parameter> <Token text=")" /> </CallInfo> </Access> <Token text=";" /></pre>
IF ENO = #c THEN ...	<pre> <Token text="IF" /> <Blank /> <Access Scope="PredefinedVariable"> <PredefinedVariable Name="ENO" /> </Access> <Blank /> <Token Text="=" /> <Blank /> <Access Scope="LocalVariable"> <Symbol> <Component Name="c" /> </Symbol> </Access> <Blank /> <Token Text="THEN" /></pre>

8.5.1.8 Export/Import multilingual comments in SCL

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)

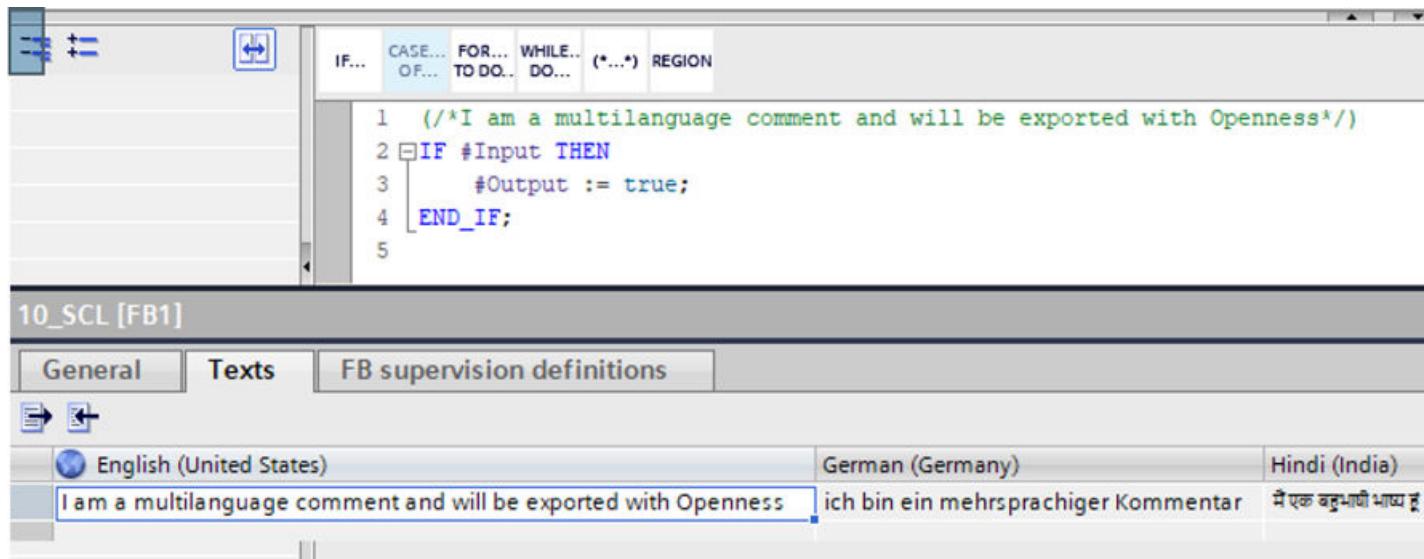
Application

It is possible in TIA Portal Openness to support import and export multilingual comments in SCL editor by the following ways:

- During SCL export in Openness, all the multilingual comments with its translation, according to enabled project languages, are exported into the same openness xml file.
- During SCL import in Openness, all the multilingual comments with its translation, according to enabled project languages, shall be imported back from the openness xml file. It shall be reflected in the project text area. The language gets enabled in if the language is not enabled in the project during import.
- After import, it shall be verified that all multilingual comment in SCL editor is linked to the appropriate comments in the project text.

Example

The following figure shows an example of Multilingual comment in SCL Editor:



The screenshot shows the TIA Portal Openness SCL Editor interface. The top menu bar includes icons for file operations and a toolbar with buttons for creating new files, opening existing ones, and saving. Below the toolbar is a menu bar with tabs: IF..., CASE... OF..., FOR... TO DO..., WHILE... DO..., (*...*), and REGION. The main workspace displays the following SCL code:

```

1 /*I am a multilanguage comment and will be exported with Openness*/
2 IF #Input THEN
3     #Output := true;
4 END_IF;
5

```

The code is numbered 1 through 5. The first line is a multi-line comment starting with a double slash. The subsequent lines define an IF block with an output assignment and an END_IF statement.

Below the workspace, the title bar reads "10_SCL [FB1]". Underneath the title bar is a tab bar with three tabs: General, Texts, and FB supervision definitions. The Texts tab is currently selected. The interface then transitions into a multilingual text editor where the same comment is displayed in three languages:

English (United States)	German (Germany)	Hindi (India)
I am a multilanguage comment and will be exported with Openness	ich bin ein mehrsprachiger Kommentar	मैं एक बहुभाषी मान्य हूँ

The following figure shows an example for export of multilingual comment in openness XML file:

8.5 Importing/exporting data of a PLC device

```
<Comment Inserted="true" UID="21">
  <MultiLanguageText Lang="en-US">I am a multilanguage comment and will be exported with Openness</MultiLanguageText>
  <MultiLanguageText Lang="de-DE">ich bin ein mehrsprachiger Kommentar</MultiLanguageText>
  <MultiLanguageText Lang="hi-IN">मैं एक बहुभाषी भाष्य हूँ</MultiLanguageText>
</Comment>
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

8.5.1.9 Exporting failsafe blocks

Exporting failsafe blocks

It is now possible to import and export the Failsafe blocks, but F-system blocks cannot be exported.

Importing failsafe blocks

Consistent F-blocks can be exported and re-imported. They will be created as F-blocks.

It will be imported as standard block if you remove the prefixes "F_" from the value of all attributes "ProgrammingLanguage".

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

[Exporting blocks \(Page 868\)](#)

[Exporting blocks with know-how protection \(Page 826\)](#)

8.5.1.10 Exporting system blocks

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)
- The project contains a system block.
- The system block is not a F-block
- PLC is not online.

Application

Only visible system blocks will be available in the blocks composition, e.g. no SFBs or SFCs. The resulting XML file is similar to the export file of a block.

Program code

Modify the following program code to export the visible data of a block to an XML file:

```
//Exports system blocks
private static void ExportSystemBlocks(PlcSoftware plcsoftware)
{
    PlcSystemBlockGroup sbSystemGroup = plcsoftware.BlockGroup.SystemBlockGroups[0];
    foreach (PlcSystemBlockGroup group in sbSystemGroup.Groups)
    {
        foreach (PlcBlock block in group.Blocks)
        {
            block.Export(new FileInfo(string.Format(@"D:\Samples\{0}.xml", block.Name)), ExportOptions.WithDefaults);
        }
    }
}
```

8.5.1.11 Exporting GRAPH blocks with multi-language text

XML structure of GRAPH blocks with multi-language text

The export XML for GRAPH blocks contains the translated step names and transition names of the GRAPH. These translated multi-language text are represented as StepName and TransitionName elements under the parent element Step and Transition respectively. These elements contain one MultiLanguageText element for each supported language. The texts for the languages which are not set explicitly are not exported. If no translation is made, the StepName and TransitionName elements are not exported. The StepName and TransitionName elements are optional. The TIA Portal Openness XML import operation throws a recoverable exception for the graph versions < V5.0.

Example for StepName element

```
<Steps>
<Step Number="1" Init="true" Name="Step1" MaximumStepTime="T#10S" WarningTime="T#7S">
  <StepName>
    <MultiLanguageText Lang="de-DE">stepDE</MultiLanguageText>
    <MultiLanguageText Lang="en-US">stepEN</MultiLanguageText>
    <MultiLanguageText Lang="it-CH">stepIT</MultiLanguageText>
  </StepName>
  ..
</Step>
..
</Steps>
```

Example for TransitionName element

```
<Transitions>
<Transition IsMissing="false" Name="Trans1" Number="1" ProgrammingLanguage="LAD">
  <TransitionName>
    <MultiLanguageText Lang="de-DE">transDE</MultiLanguageText>
    <MultiLanguageText Lang="en-US">transEN</MultiLanguageText>
    <MultiLanguageText Lang="it-CH">transIT</MultiLanguageText>
  </TransitionName>
  ..
</Transition>
..
</Transitions>
```

8.5.1.12 Importing block

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- PLC is not online.

Application

The TIA Portal Openess API supports the import of blocks with "LAD", "FBD", "GRAPH", "SCL" or "STL" programming languages from an XML file. The following block types are supported:

- Function blocks (FB)
- Functions (FC)
- Organization blocks (OB)
- Global data blocks (DB)

Note

Importing optimized data blocks

Optimized data blocks are only supported by CPUs as of S7-1200. If you import optimized data blocks into S7-300 or S7-400, an exception is thrown and the import fails.

Response to importing

The following rules apply when importing a block:

- The XML file can contain less data than the block in the project, e.g., fewer parameters.
- Redundant information, such as call information, must be identical in the project and in the XML file. Otherwise, an exception is thrown.
- The data in the XML file may be "inconsistent" regarding their ability to be compiled in the TIA Portal.
- Attributes with the attributes "ReadOnly=True" and "Informative=True" are not imported.
- Missing instance DBs are not created automatically.
- If no block number is specified in the xml file, the block number is assigned automatically.
- If the block is not existing in the project and no version information is specified in the xml file, the version "0.1" is assigned.

Program code

Modify the following program code:

```
//Import blocks
private static void ImportBlocks(PlcSoftware plcSoftware)
{
    PlcBlockGroup blockGroup = plcSoftware.BlockGroup;
    IList<PlcBlock> blocks = blockGroup.Blocks.Import(new FileInfo(@"D:\Blocks
\myBlock.xml"), ImportOptions.Override);
}
```

8.5 Importing/exporting data of a PLC device

Modify the following program code:

```
//Import system blocks
private static void ImportSystemBlocks(PlcSoftware plcSoftware)
{
    PlcBlockSystemGroup systemblockGroup = plcSoftware.BlockGroup;
    IList<PlcBlock> blocks = systemblockGroup.Blocks.Import(new FileInfo(@"D:\Blocks
\myBlock.xml"), ImportOptions.Override);
}
```

8.5.1.13 Exporting blocks

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- PLC is not online.

Application

The API interface supports exporting consistent blocks and user data types to an XML file.

The XML file receives the name of the block. The following block types are supported:

- Function blocks (FB)
- Functions (FC)
- Organization blocks (OB)
- Global data blocks (DB)

The following programming languages are supported:

- STL
- FBD
- LAD
- GRAPH
- SCL

Attributes applicable for all blocks

The following attributes are exported in all blocks with the selected `ExportOptions` (see Exporting configuration data (Page 752)). Attributes in bold typeface are always exported.

Additional information is available in the TIA Portal information system under "Overview of block attributes".

Attribute	Type	Default value	ReadOnly
AutoNumber	Bool	true	false
CodeModifiedDate	DateTime	-	true
CompileDate	DateTime	-	true
CreationDate	DateTime	-	true
HeaderAuthor	String	""	false
HeaderFamily	String	""	false
HeaderName	String	""	false
HeaderVersion	String	"0.1"	false
Interface	String	empty interface	false
InterfaceModifiedDate	DateTime	-	true
IsConsistent	Bool	-	true
IsKnowHowProtected ¹	Bool	false	true
IsWriteProtected	Bool	false	true
MemoryLayout	enum MemoryLayout	-	false
ModifiedDate	DateTime	-	true
Name	String	-	false
Number	Int32	next available number	false
ParameterModified	DateTime	-	true
PLCSimAdvancedSupport	Bool	false	true
ProgrammingLanguage	enum ProgrammingLanguage	-	false
StructureModified	DateTime	-	true

¹ The IsKnowHowProtected attribute is applicable for UDT too.

Note

There are certain conditions where MemoryLayout attribute can be readonly.

Dynamic accessible general attributes

The following attributes which can be read only in some conditions:

Attribute	ReadOnly condition
AutoNumber	All KnowHowProtected blocks, All Classic OBs; Plus OBs: DiagnosticErrorInterrupt, IOAccessError, ProgrammingError, PullOrPlugOfModules, RackOrStationFailure, Status, TimeErrorInterrupt, Update
HeaderVersion	System- and KnowHowProtected DBs; ArrayDBs that originated from system library
HeaderName	
HeaderFamily	
HeaderAuthor	
MemoryLayout	Classic blocks, System Know how protected blocks, ArrayDBs, IDBofFBs, Graph blocks

Attributes applicable for ArrayDB block

The following attributes are exported for ArrayDB block with the selected ExportOptions.

Attribute	Type	Default value	ReadOnly
ArrayDataType	String	-	true
ArrayLimitUpperBound	Int32	-	true

Attributes applicable for DB block

The following attributes are exported in DB block with the selected ExportOptions.

Attribute	Type	Default value	ReadOnly
IsOnlyStoredInLoadMemory	Bool	false	false
IsPLCDB	Bool	false	false
IsWriteProtectedInAS	Bool	false	false

Attributes applicable for FB block

The following attributes are exported for FB blocks with the selected ExportOptions.

Attribute	Type	Default value	ReadOnly
AssignedProDiagFB	String	-	-
ISMultInstanceCapable	Bool	-	true
Supervisions	String	no supervisions	true for IDB of FB and false for FB

Attributes applicable for DB and FB blocks

The following attributes are exported in DB and FB block with the selected ExportOptions.

Attribute	Type	Default value	ReadOnly
IsIECCheckEnabled	Bool	false	false
IsRetainMemResEnabled ¹	Bool	false	false
MemoryReserve	Unsigned	0	false
RetainMemoryReserve ²	Unsigned	0	false

² If the "IsRetainMemResEnabled" attribute's value is "false", and the "RetainMemoryReserve" attribute is not equal to "0", an exception is thrown.

Attributes applicable for FB, DB and IDB blocks

The following attributes are exported in FB, DB, and IDB blocks with the selected ExportOptions.

Attribute	Type	Default value	ReadOnly
DownloadWithoutReinit	Bool	false	true

Attributes applicable for FB and FC blocks

The following attributes are exported for FB and FC block with the selected ExportOptions.

Attribute	Type	Default value	ReadOnly
LibraryType	String	-	true
LibraryTypeVersionGuid	String	-	true

Attributes applicable for FB and FC (STL) blocks

The following attributes are exported for FB and FC (STL) block with the selected ExportOptions.

Attribute	Type	Default value	ReadOnly
ParameterPassing	Bool	false	false

Attributes applicable for FB, FC and instance DB of an FB block

The following attributes are exported for FB, FC and instance DB of an FB block with the selected ExportOptions.

Attribute	Type	Default value	ReadOnly
UDABlockProperties	String	""	false
UDAEnableTagReadback	Bool	false	false

Attributes applicable for instance DB of FB and UDT

The following attributes are exported for instance DB of FB and UDT block with the selected ExportOptions.

Attribute	Type	Default value	ReadOnly
InstanceOfName	String	""	false
InstanceOfNumber	Unsigned Short	-	true
InstanceOfType	enum BlockType	-	true

Attribute	Type	Default value	ReadOnly
OfSystemLibElement	String	""	false
OfSystemLibVersion	String	""	false

Attributes applicable for OB block

The following attributes are exported in OB block for specific Plus PLCs with the selected ExportOptions.

Attribute	Type	Default value	ReadOnly
ApplicationCycle	Single	-	true
AutomaticMinimum	Bool	-	true
ConstantName	String	-	true
CycleTimeDistributedIO	Single	-	true
CyclicApplicationCycleTime	Single	-	true
CyclicTime	Int32	100000	true
DataExchangeMode	OBDataExchangeMode	Cyclic	true
DelayTime	Double	-	true
DistributedIOPortName	String	-	true
DistributedIOPortNumber	Int32	-	true
EnableTimeError	Bool	-	true
EventClass	String	-	true
EventsToBeQueued	Int32	-	true
EventThresholdForTimeError	Int32	-	true
Execution	OBExecution	Never	true
Factor	Single	-	true
PhaseOffset	Int32	0	true
PriorityNumber	Int32	-	true
ProcessImagePartNumber	UInt32	-	true
ReportEvents	Bool	-	true
SecondaryType ³	String	-	false
StartDate	DateTime	1/1/2012	true
SynchronousApplicationCycleTime	Single	-	true
TimeMode	OBTimeMode	System	true
TimeOfDay	DateTime	12:00 AM	true
TransformationDBNumber	UInt16	0xffff	true

³ When exporting an OB, the "SecondaryType" is additionally set based on the OB number. The assignment is checked during import. If the assignment is incorrect, an exception of type "Recoverable" is thrown.

Attributes applicable for FB, FC and OB blocks

The following attributes are exported for FB, FC and OB block with the selected ExportOptions.

Attribute	Type	Default value	ReadOnly
HandleErrorsWithinBlock	Bool	false	true

Attributes applicable for FB, FC and UDT blocks

The following attributes are exported for FB, FC and UDT block with the selected ExportOptions.

Attribute	Type	Default value	ReadOnly
LibraryConformanceStatus	String	-	false

Attributes applicable for GRAPH block

The following attributes are exported for GRAPH block with the selected ExportOptions.

Attribute	Type	Default value	ReadOnly
AcknowledgeErrorsRequired	Bool	true	false
CreateMinimizedDB	Bool	false	false
ExtensionBlockName	String	-	-
GraphVersion	String	-	false
InitialValuesAcquisition	String	-	-
LanguageInNetworks	String	-	false
LockOperatingMode	Bool	false	false
PermanentILProcessingIn-MANMode	Bool	false	false
SkipSteps	Bool	false	false

Attributes applicable for GRAPH FB block

The following attributes are exported for GRAPH FB block with the selected ExportOptions.

Attribute	Type	Default value	ReadOnly
WithAlarmHandling	Bool	true	false

Attributes applicable for SCL block

The following attributes are exported for SCL blocks with the selected ExportOptions. These attributes are exported based on the type of PLCs.

Attribute	Type	Default value	ReadOnly
CheckArrayLimits	Bool	false	false
ExtendedStatus	Bool	false	false
DBAccessibleFromOPCUA	Bool	true	false

Attributes applicable for GRAPH, SCL, and LAD/FBD blocks

The following attributes are exported for GRAPH, SCL, and LAD/FBD blocks with the selected ExportOptions.

Attribute	Type	Default value	ReadOnly
SetENOAutomatically	Bool	-	false

Attributes applicable for program blocks (except OB), DB, and UDT blocks

The following attributes are exported for program blocks (except OB), DB, and UDT blocks under a unit with the selected ExportOptions.

Attribute	Type	Default value	ReadOnly
Access	UnitAccessType	Unpublished	false

For Import behavior for the attribute 'Access'

	Imported under unit	Imported not under unit (without SWImportOptions.IgnoreUnitAttributes)	Imported not under unit (with SWImportOptions.IgnoreUnitAttributes)
XML exported from under unit	'Access' used and set	RecoverableException is thrown	'Access' ignored
XML exported from not under unit	'Access' gets its default value	'Access' doesn't exist	'Access' doesn't exist

Program code

Modify the following program code to export a block without know-how protection to an XML file:

```
//Exports a regular block
private static void ExportRegularBlock(PlcSoftware plcSoftware)
{
    PlcBlock plcBlock = plcSoftware.BlockGroup.Blocks.Find("MyBlock");
    plcBlock.Export(new FileInfo(string.Format(@"D:\Samples\{0}.xml", plcBlock.Name)), ExportOptions.WithDefaults);
}
```

8.5.1.14 Importing blocks/UDT with open reference

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- PLC is not online

Application

Using Openness API, you can use a new import mode for STEP7 objects to import blocks and UDTs even if a related object is missing.

The Openness interface supports the new import mode for the following conditions:

Import of	Object reference
UDT	UDT
DB (global)	UDT
IDBofUDT	UDT
IDBofFB	FB
ArrayDB	Array of UDT
FB	UDT (interface), Multi-instance
FC	UDT (Interface)

Program code

You can use the new mode by a new overload of the respective Import method. The new overload has an additional parameter which accepts a value of the new flagged enum SWImportOptions. To allow the import, you can use SWImportOptions.IgnoreMissingReferencedObject, even if the referenced object is missing.

```
Flagged Enum SWImportOptions
{
    None = 0,
    IgnoreStructuralChanges = 1,
    IgnoreMissingReferencedObjects = 2
}
... // All kinds of blocks
PlcBlockComposition.Import(file, ImportOptions.None,
SWImportOptions.IgnoreMissingReferencedObject);
...
... // UDTs
PlcTypeComposition.Import(file, ImportOptions.None,
SWImportOptions.IgnoreMissingReferencedObject);
...
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

8.5.1.15 Importing blocks/UDT for structural change object

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open
See [Opening a project \(Page 106\)](#)
- PLC is not online

Application

Using Openness API, you can import blocks and UDTs even if instance data is lost because of a structural change of related objects.

The Openness interface supports the new import mode for the following conditions:

Import of	Object references
Tag	UDT
UDT	UDT

Import of	Object references
DB (global)	UDT
IDBofUDT	UDT
IDBofFB	FB
ArrayDB	Array of UDT
FB	UDT (interface), Multi-instance
FC	UDT (Interface)

Program code

You can use new mode by a new overload of the respective Import method. The new overload has an additional parameter which accepts a value of the new flagged enum SWImportOptions. To allow the import although there are structural change and maybe a data loss using "SWImportOptions.IgnoreStructuralChanges".

```
Flagged Enum SWImportOptions
{
    None = 0,
    IgnoreStructuralChanges = 1,
    IgnoreMissingReferencedObjects = 2
}
...
// All kinds of blocks
PlcBlockComposition.Import(file, ImportOptions.None,
SWImportOptions.IgnoreStructuralChanges);
...
...
// UDTs
PlcTypeComposition.Import(file, ImportOptions.None,
SWImportOptions.IgnoreStructuralChanges);
...
```

See also

- Connecting to the TIA Portal (Page 76)
- Opening a project (Page 106)

8.5.1.16 Export/Import of unit specific publishing attribute of blocks and types

Requirement

- The TIA Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)

Application

The 'Access' attribute exists only for blocks, plc types, and tag tables located under a unit. therefore the exported XML will contain it. The exported XML of the same block, type, or tag table located in non-unit environment won't contain it.

The general Openness XML import rules won't allow importing XMLs containing unknown/undefined attributes. Because of this rule the exported XMLs originated from object in a unit environment cannot be imported in a non-unit environment. This would be a huge limitation in the usage therefore a following extension is defined.

The import () method has an overload to pass three parameters. The following are the these parameters:

Parameter	Return type	Description
path	String	Specifies the path to the Simatic ML file to be imported
importOptions	enum: Siemens.Engineering.ImportOptions	Specifies the general import option(s) to be used during import.
swImportOptions	enum: Siemens.Engineering.SW.SWImportOptions	Specifies the import option(s) specific to Step7 to be used during import.

The enum type Siemens.Engineering.SW.SWImportOptions will be extended with the following new import option:

- IgnoreUnitAttributes: specifies that the import process shouldn't be aborted in case of unit related attributes are present in the XML and the import is performed in a non-unit environment.

The 'IgnoreUnitAttributes' is only taken into consideration when the XML

- Is exported from a unit
- contains the 'Access' attribute
- Is imported into non-unit environment

If the exported XML doesn't contain the 'Access' Openness attribute and / or it is imported into a unit the new import option won't be considered by the import logic at all.

Program code:

```

PlcSoftware plcTarget = GetControllerTargetByPLCName(Session.OpnsProject.Devices, PLCName);
PlcUnitProvider plcUnitProvider = plcTarget.GetService<PlcUnitProvider>();
PlcSoftware plcSoftware = plcTarget.GetService<SoftwareContainer>() as PlcSoftware;
PlcUnit plcUnit1 = plcUnitProvider.UnitGroup.Units[0];
//assuming Unit_1 is already existing
PlcUnit plcUnit2 = plcUnitProvider.UnitGroup.Units[1];
//assuming Unit_2 is already existing
PlcBlock block1 = plcUnit1.BlockGroup.Blocks.Find("Block_1");
//assuming Block_1 is already existing under Unit_1
PlcBlock block2 = plcUnit2.BlockGroup.Blocks.Find("Block_2");
//assuming Block_2 is already existing under Unit_2
PlcBlock block3 = plcSoftware.BlockGroup.Blocks.Find("Block_3");
//assuming Block_3 is already existing under the PLC

```

Program code: Object exported from a unit and imported into a unit

The examples below are demonstrating the behavior of the new import option by using blocks, the same applies for plc types and tag tables as well.

Modify the following program code to export a block with 'Access' attribute set to value 'Unpublished' (default value) with ExportOptions.WithDefaults and import with SWImportOptions.None:

```

block1.SetAttribute("Access", UnitAccessType.Unpublished);
block1.Export(new FileInfo("somepath"), ExportOptions.WithDefaults);
block1.Delete();
plcUnit2.BlockGroup.Blocks.Import(new FileInfo("somepath"), ImportOptions.None,
SWImportOptions.None);

```

Modify the following program code to export a block with 'Access' attribute set to value 'Unpublished' (default value) with ExportOptions.WithDefaults and import with SWImportOptions.IgnoreUnitAttributes:

```

block1.SetAttribute("Access", UnitAccessType.Unpublished);
block1.Export(new FileInfo("somepath"), ExportOptions.WithDefaults);
block1.Delete();
plcUnit2.BlockGroup.Blocks.Import(new FileInfo("somepath"), ImportOptions.None,
SWImportOptions.IgnoreUnitAttributes);

```

Modify the following program code to export a block with 'Access' attribute set to value 'Published' with ExportOptions.None and import with SWImportOptions.None:

```

block1.SetAttribute("Access", UnitAccessType.Published);
block1.Export(new FileInfo("somepath"), ExportOptions.None);
block1.Delete();
plcUnit2.BlockGroup.Blocks.Import(new FileInfo("somepath"), ImportOptions.None,
SWImportOptions.None);

```

8.5 Importing/exporting data of a PLC device

Modify the following program code to export a block with 'Access' attribute set to value 'Published' with ExportOptions.None and import with SWImportOptions.IgnoreUnitAttributes:

```
block1.SetAttribute("Access", UnitAccessType.Published);  
block1.Export(new FileInfo("somepath"), ExportOptions.None);  
block1.Delete();  
plcUnit2.BlockGroup.Blocks.Import(new FileInfo("somepath"), ImportOptions.None,  
SWImportOptions.IgnoreUnitAttributes);
```

Note

In all of the above program codes, No exception is thrown, and import succeeds.

Program code: Object exported from a unit and imported into non-unit environment

Modify the following program code to export a block with 'Access' attribute set to value 'Unpublished' (default value) with ExportOptions.WithDefaults and import with SWImportOptions.None:

```
block1.SetAttribute("Access", UnitAccessType.Unpublished);  
block1.Export(new FileInfo("somepath"), ExportOptions.WithDefaults);  
block1.Delete();  
plcUnit2.BlockGroup.Blocks.Import(new FileInfo("somepath"), ImportOptions.None,  
SWImportOptions.None);
```

Note

In the above program code, Import does not succeed and a Recoverable Exception is thrown.

Modify the following program code to export a block with 'Access' attribute set to value 'Unpublished' (default value) with ExportOptions.WithDefaults and import with SWImportOptions.IgnoreUnitAttributes:

```
block1.SetAttribute("Access", UnitAccessType.Unpublished);  
block1.Export(new FileInfo("somepath"), ExportOptions.WithDefaults);  
block1.Delete();  
plcUnit2.BlockGroup.Blocks.Import(new FileInfo("somepath"), ImportOptions.None,  
SWImportOptions.IgnoreUnitAttributes);
```

Note

In the above program code, import succeed and No exception is thrown.

Modify the following program code to export a block with 'Access' attribute set to value 'Published' with ExportOptions.None and import with SWImportOptions.None:

```
block1.SetAttribute("Access", UnitAccessType.Published);
block1.Export(new FileInfo("somepath"), ExportOptions.None);
block1.Delete();
plcUnit2.BlockGroup.Blocks.Import(new FileInfo("somepath"), ImportOptions.None,
SWImportOptions.None);
```

Note

In the above program code, Import succeed and a Recoverable exception is thrown.

Modify the following program code to export a block with 'Access' attribute set to value 'Published' with ExportOptions.None and import with SWImportOptions.IgnoreUnitAttributes:

```
block1.SetAttribute("Access", UnitAccessType.Published);
block1.Export(new FileInfo("somepath"), ExportOptions.None);
block1.Delete();
plcUnit2.BlockGroup.Blocks.Import(new FileInfo("somepath"), ImportOptions.None,
SWImportOptions.IgnoreUnitAttributes);
```

Note

In the above program code, No exception is thrown and import succeed.

Object exported from non-unit environment into a unit

The exported XML doesn't contain the 'Access' Openness attribute and when importing it gets the default value 'Unpublished'.

Object exported from non-unit environment into non-unit environment

The exported XML doesn't contain the 'Access' Openness attribute and when importing it nothing happens.

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

8.5.2 Technology objects

8.5.2.1 S7-1200 Motion Control

8.5.2.2 S7-1500 Motion Control

8.5.2.3 PID control

8.5.2.4 Counting

8.5.2.5 Easy Motion Control

8.5.3 Tag tables

8.5.3.1 Exporting PLC tag tables

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

One XML file is exported per PLC tag table.

The TIA Portal Openness API supports the export of all PLC tag tables from the system group and its subgroups.

Program code

Modify the following program code to export all PLC tag tables from the system group and its subgroups:

```
private static void ExportAllTagTables(PlcSoftware plcSoftware)
{
    PlcTagTableSystemGroup plcTagTableSystemGroup = plcSoftware.TagTableGroup;
    // Export all tables in the system group
    ExportTagTables(plcTagTableSystemGroup.TagTables);
    // Export the tables in underlying user groups
    foreach(PlcTagTableUserGroup userGroup in plcTagTableSystemGroup.Groups)
    {
        ExportUserGroupDeep(userGroup);
    }
}

private static void ExportTagTables(PlcTagTableComposition tagTables)
{
    foreach(PlcTagTable table in tagTables)
    {
        table.Export(new FileInfo(string.Format(@"D:\Samples\{0}.xml", table.Name)), ExportOptions.WithDefaults);
    }
}

private static void ExportUserGroupDeep(PlcTagTableUserGroup group)
{
    ExportTagTables(group.TagTables);
    foreach(PlcTagTableUserGroup userGroup in group.Groups)
    {
        ExportUserGroupDeep(userGroup);
    }
}
```

See also

[Exporting configuration data \(Page 752\)](#)

8.5.3.2 Importing PLC tag table

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)

Program code

Modify the following program code to import PLC tag tables or a folder structure with PLC tag tables from an XML file into the system group or a user-defined group:

```
//Imports tag tables to the tag system group
private static void ImportTagTable(PlcSoftware plcSoftware)
{
    PlcTagTableSystemGroup plcTagTableSystemGroup = plcSoftware.TagTableGroup;
    PlcTagTableComposition tagTables = plcTagTableSystemGroup.TagTables;
    tagTables.Import(new FileInfo(@"D:\Samples\myTagTable.xml"), ImportOptions.Override);
    // Or, to import into a subfolder:
    // plcTagTableSystemGroup.Groups.Find("SubGroup").TagTables.Import(new
    FileInfo(@"D:\Samples\myTagTable.xml"), ImportOptions.Override);
}
```

See also

[Notes on performance of TIA Portal Openness \(Page 44\)](#)

8.5.3.3 Exporting an individual tag or constant from a PLC tag table

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
[See Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
[See Opening a project \(Page 106\)](#)

Application

The API interface supports the export of a tag or constant from a PLC tag table to an XML file. Make sure that the tag table names used conform to the file naming conventions of your file system.

The comment of a tag or constant is only exported if at least one language is set for the comment. If the comment is not set for all project languages, this comment is only exported for the set project languages.

Note

PLC system constants

PLC system constants are excluded from export and import.

Program code

Modify the following program code to export a specific tag or constant from a PLC tag table to an XML file:

```
//Exports a single tag or constant of a controller tag table
private static void ExportTag(PlcSoftware plcSoftware, string tagName)
{
    PlcTagTableSystemGroup plcTagTableSystemGroup = plcSoftware.TagTableGroup;
    PlcTag tag = plcTagTableSystemGroup.TagTables[0].Tags.Find(tagName);
    if(tag == null) return;

    tag.Export(new FileInfo(string.Format(@"D:\Samples\{0}.xml", tag.Name)),
    ExportOptions.WithDefaults);
}

private static void ExportUserConstant(PlcSoftware plcSoftware, string userConstantName)
{
    PlcTagTableSystemGroup plcTagTableSystemGroup = plcSoftware.TagTableGroup;
    PlcUserConstant plcConstant =
plcTagTableSystemGroup.TagTables[0].UserConstants.Find(userConstantName);
    if(plcConstant== null) return;

    plcConstant.Export(new FileInfo(string.Format(@"D:\Samples\{0}.xml",
plcConstant.Name)), ExportOptions.WithDefaults);
}
```

See also

[Exporting configuration data \(Page 752\)](#)

[Notes on performance of TIA Portal Openness \(Page 44\)](#)

8.5.3.4 Importing an individual tag or constant into a PLC tag table

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)

Application

You can import either tags or constants in an import call.

Note

Constants can only be imported as user constants.

Program code

Modify the following program code to import tag groups or individual tags and constants from an XML file:

```
//Imports tags into a plc tag table
private static void ImportTag(PlcSoftware plcSoftware, string tagtableName)
{
    PlcTagTableSystemGroup plcTagTableSystemgroup = plcSoftware.TagTableGroup;
    PlcTagTable tagTable = plcTagTableSystemgroup.TagTables.Find(tagtableName);
    if(tagTable == null) return;

    tagTable.Tags.Import(new FileInfo(@"D:\Samples\myTags.xml"), ImportOptions.Override);
}

//Imports constants into a plc tag table
private static void ImportConstant(PlcSoftware plcSoftware, string tagtableName)
{
    PlcTagTableSystemGroup plcTagTableSystemgroup = plcSoftware.TagTableGroup;
    PlcTagTable tagTable = plcTagTableSystemgroup.TagTables.Find(tagtableName);
    if(tagTable == null) return;

    tagTable.UserConstants.Import(new FileInfo(@"D:\Samples\myConstants.xml"),
ImportOptions.Override);
}
```

See also

[Exporting configuration data \(Page 752\)](#)

[Notes on performance of TIA Portal Openness \(Page 44\)](#)

8.5.4 Exporting user data type

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)
- PLC is not online

Program code

Modify the following program code to export an user data type to an XML file:

```
//Exports a user defined type
private static void ExportUserDefinedType(PlcSoftware plcSoftware)
{
    string udtname = "udt name XYZ";
    PlcTypeComposition types = plcSoftware.TypeGroup.Types;
    PlcType udt = types.Find(udtname);
    udt.Export(new FileInfo(string.Format(@"C:\OpennessSamples\udts\{0}.xml", udt.Name)), 
ExportOptions.WithDefaults);
}
```

8.5.5 Importing user data type

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
Connecting to the TIA Portal (Page 76)
- A project is open.
Opening a project (Page 106)
- PLC is not online.

Application

The API interface supports the importing of user data types from an XML file.

Import file syntax

The following code example shows an excerpt from an import file of a user-defined data type:

```
<Section Name="Input">
<Member Name="Input1" Datatype="myudt1">
<Sections>
<Section Name="None">
<Member Name="MyUDT1Member1" Datatype="bool"/>
<Member Name="MyUDT1Member2" Datatype="myudt1">
<Sections...>
```

Note

Syntax for user-defined data types of elements

An exception is thrown if the user-defined data type of an element in the import file for user data types has incorrect syntax.

Make sure that user-defined data types are noted with ";.

Program code

Modify the following program code to import a user data type:

```
//Imports user data type
private static void ImportUserDataType(PlcSoftware plcSoftware)
{
    FileInfo fullFilePath = new FileInfo(@"C:\OpennessSamples\Import\ExportedPlcType.xml");
    PlcTypeComposition types = plcSoftware.TypeGroup.Types;
    IList<PlcType> importedTypes = types.Import(fullFilePath, ImportOptions.Override);
}
```

See also

[Importing configuration data \(Page 753\)](#)

8.5.6 Export of data in OPC UA XML format

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)
- PLC is not online

Application

You can use the TIA Portal Openness to export PLC data as OPC UA XML file. As input parameter for the action you need an absolute directory path, where the xml file will be saved.

Program code

Modify the following program code to export PLC data as OPC UA XML file:

```
//Export PLC data as OPC UA XML file
private static void OpcUaExport(Project project, DeviceItem plc)
{
OpcUaExportProvider opcUaExportProvider = project.HwUtilities.Find("OPCUAExportProvider")
as OpcUaExportProvider;
if (opcUaExportProvider == null) return;
opcUaExportProvider.Export(plc, new FileInfo(string.Format(@"D:\OPC UA export files
\{0}.xml", plc.Name)));
}
```

8.6 Importing/exporting hardware data

8.6.1 AML file format

Introduction

AutomationML is a neutral data format based on XML for the storage and exchange of plant engineering information, which is provided as open standard. Goal of AutomationML is to interconnect the heterogeneous tool landscape of modern engineering tools in their different disciplines, e.g. mechanical plant engineering, electrical design, HMI, PLC, robot control.

The class model used for the export and import of CAx data is based on the following AML standards:

- Whitepaper AutomationML Part 1 – AutomationML Architecture, October 2014
- Whitepaper AutomationML Part 2 – AutomationML Role Libraries, October 2014
- Whitepaper AutomationML Part 4 – AutomationML Logic, May 2010
- Whitepaper AutomationML – AutomationML Communication, September 2014
- Whitepaper AutomationML – AutomationML and eCI@ss Integration, November 2015
- Application Recommendations: Automation Project Configuration - AR_001E Version 1.0.0, Mai.2017

Schema

The AutomationML data exchange model is described by the CAEX schema Version 2.15. You can download this file from the homepage of AutomationML e.V. (<https://www.automationml.org/o.red.c/dateien.html>)

8.6.2 Pruned AML

Introduction

Pruning is the act of optimizing the content by removing certain things which are not necessarily to be provided. In case of external tools like EPLAN , the auto created sub module information within a hardware configuration has no significance with respect to EPLAN. Hence, these tools generate an AML file by removing the auto created sub module information from the hardware configuration. This file is called as pruned AML.

Generation of pruned AML

Generation of a Pruned AML is based on the following rules in order.

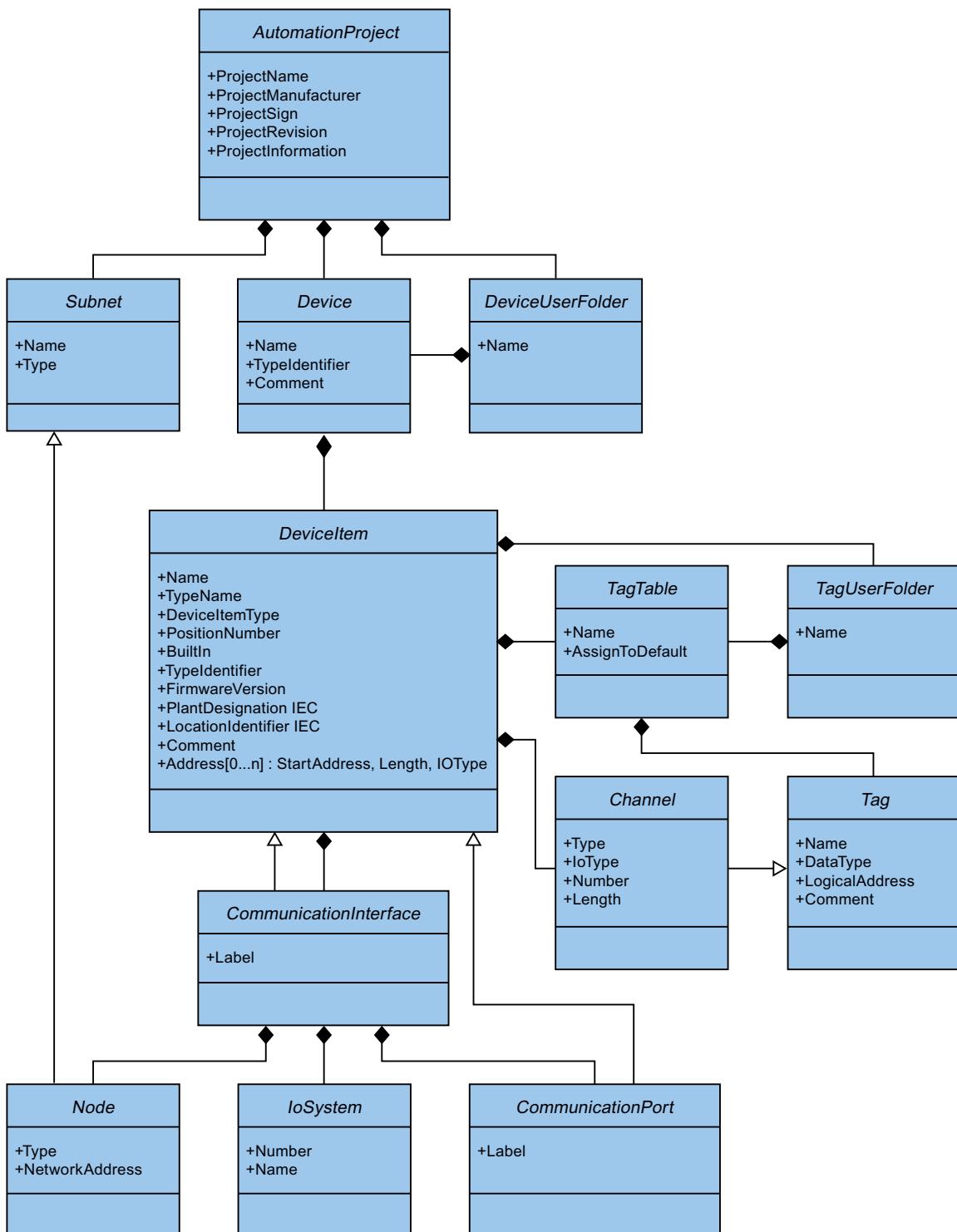
1. If a device item is pluggable, it shall not be pruned.
2. If a device item is of type "interface" or "port", it shall not be pruned.
3. If a device item is built-in and is at rack level, it shall not be pruned
4. AddressObjects of type "diagnosis" are not relevant for the prune algorithm.
5. Address objects linked with the auto created sub modules shall be provided under the immediate parent (which shall be a non-auto created sub module).
6. Address objects shall be included in the same sequence as returned by TIA Portal Openness.

8.6.3 Overview of the objects and parameters of the CAx import/export

Export/import objects and attributes

The following figure shows the exportable objects with their attributes and dependencies of the CAx Import/Export.

8.6 Importing/exporting hardware data



8.6.4 Structure of the CAx data for importing/exporting

Basic structure of an export file

The data in the export file from the import/export is structured with reference to a basic structure. The export file is generated in an AML format.

The AML file starts with a document information. The file includes the data of the computer-specific installation of the exported project.

It shall be possible in TIA Portal Openess V16 onwards to support export and import AML files with a new version of "AR APC V1.1.0".

TIA Portal V16 can import AML files with "AR APC V1.0.0" and "AR APC V1.1.0".

AML files generated by TIA Portal older versions like V14 SP1, V15, V15.1 should still export and import AML files "AR APC V1.0.0".

The export file comprises the following two sections:

- Additional information

The <WriterHeader> includes information about the export or import process. The import ignores the content of the <AdditionalInformation> section.

For example you can insert a <AdditionalInformation>...</AdditionalInformation> block, in which you place additional information about the validation. After the AML file is forwarded, the recipient can use this block before the import to check whether the AML file has been changed.

```
<xml version="1.0" encoding="utf-8">
<CAEXFile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    FileName="CAx_asterisk_AML_03_V14.aml" SchemaVersion="2.15"
    xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd">
<AdditionalInformation>
    <WriterHeader>
        <WriterName>Totally Integrated Automation Portal</WriterName>
        <WriterID>1d4fccebb-1ad6-4881-b01d-bca335d94a46:V1.0</WriterID>
        <WriterVendor>Siemens AG</WriterVendor>
        <WriterVendorURL>www.siemens.com</WriterVendorURL>
        <WriterVersion>1400</WriterVersion>
        <WriterRelease>1400.100.101.16</WriterRelease>
        <LastWritingDateTime>2016-09-29T11:21:34.9551066Z</LastWritingDateTime>
    </WriterHeader>
</AdditionalInformation>
<AdditionalInformation AutomationMLVersion="2.0" />
<AdditionalInformation DocumentVersions="Recommendations">
    <Document DocumentIdentifier="AR APC" Version="1.1.0" />
</AdditionalInformation>
...
...
```

Note

CAx shall perform export and import of AR APC version in AML file in accordance to the installed version of TIA Portal.

- Instance hierarchy

This section contains the hierarchical sequence of the exported internal elements.

```

...
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
<InternalElement ID="d4dc896a-f4a5-41b6-9c48-8d3a0a5a4343" Name="CAx_asterisk_AML_03_V14">
  <Attribute Name="ProjectManufacturer" AttributeDataType="xs:string" />
  <Attribute Name="ProjectSign" AttributeDataType="xs:string" />
  <Attribute Name="ProjectRevision" AttributeDataType="xs:string" />
  <Attribute Name="ProjectInformation" AttributeDataType="xs:string" />
  <InternalElement ID="544f3a69-5f65-45ba-ac2f-1448db9493fd" Name="PN/IE_1">
    ...
  </InternalElement>

  <InternalElement ID="12116ac0-94b7-49d2-888d-7d39bbc0caf5" Name="S71500/ET200MP station_1">
    ...
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath=
    "AutomationProjectConfigurationRoleClassLib/AutomationProject" />
  <InternalLink Name="Link To Port_1" RefPartnerSideA=
    "726148ce-de5b-4728-8886-4ba273435479:CommunicationPortInterface" RefPartnerSideB=
    "cb9d24f3-8200-4c89-9b40-24ae850e293e:CommunicationPortInterface" />
  <InternalLink Name="Link To Port_2" RefPartnerSideA=
    "cb9d24f3-8200-4c89-9b40-24ae850e293e:CommunicationPortInterface" RefPartnerSideB=
    "726148ce-de5b-4728-8886-4ba273435479:CommunicationPortInterface" />
  ...
  <InternalLink Name="Link To IoSystem_3" RefPartnerSideA=
    "d8f6e006-3778-4a05-aab1-df844fe822fe:LogicalEndPoint_Interface" RefPartnerSideB=
    "2344b7af-329c-4215-92d1-6143b4627b56:LogicalEndPoint_IoSystem" />
</InternalElement>
</InstanceHierarchy>
</CAEXFile>

```

Internal elements

All objects inside the instance hierarchy of the AML file are `InternalElements`. The internal element `AutomationProject` contains all internal elements of all role classes. Every internal element supports a set of attributes.

The attribute `<TypeIdentifier>` identifies every object type of a hardware object that is creatable via TIA Portal Openness.

Note

Auto-created objects

Auto-created objects can only be created by other objects. They do not have properties or a type-identifier. They are included in the exported file, but you cannot trigger the export of a specific autocreated object.

At the end of the AML-element of an internal element, the following are defined:

- Role class

The `SupportedRoleClass` element defines the object type of the internal element. The object type is defined in the role class library that maps the standard AML to the object model of TIA Portal Openness and TIA Portal.

```
...
<InternalElement ID="1d1a37ed-19d9-4a23-bc91-51f5a8e0244b" Name="Ungrouped devices">
<InternalElement ID="ab193f5d-0375-4a6d-a576-a903e2b77cca" Name="ET 200SP station_1">
...
<InternalElement ID="72d41729-90a7-4de3-9708-a8eeda6b1886" Name="IO device_1">
...
<SupportedRoleClass RefRoleClassPath=
  "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
</InternalElement>
...
<SupportedRoleClass RefRoleClassPath=
  "AutomationProjectConfigurationRoleClassLib/Device" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath=
  "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
</InternalElement>
...

```

- Internal link

The element `InternalLink` defines the communication partners of a connection.

```
...
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
<InternalElement ID="d4dc896a-f4a5-41b6-9c48-8d3a0a5a4343" Name="CAx_asterisk_AML_03_V14">
<Attribute Name="ProjectManufacturer" AttributeDataType="xs:string" />
<Attribute Name="ProjectSign" AttributeDataType="xs:string" />
<Attribute Name="ProjectRevision" AttributeDataType="xs:string" />
<Attribute Name="ProjectInformation" AttributeDataType="xs:string" />
...
<SupportedRoleClass RefRoleClassPath=
  "AutomationProjectConfigurationRoleClassLib/AutomationProject" />
<InternalLink Name="Link To Port_1" RefPartnerSideA=
  "726148ce-de5b-4728-8886-4ba273435479:CommunicationPortInterface" RefPartnerSideB=
  "cb9d24f3-8200-4c89-9b40-24ae850e293e:CommunicationPortInterface" />
<InternalLink Name="Link To Port_2" RefPartnerSideA=
  "cb9d24f3-8200-4c89-9b40-24ae850e293e:CommunicationPortInterface" RefPartnerSideB=
  "726148ce-de5b-4728-8886-4ba273435479:CommunicationPortInterface" />
<InternalLink Name="Link To Port_3" RefPartnerSideA=
  "65307e3e-95fd-41ac-9982-e5e4fffc2fb15:CommunicationPortInterface" RefPartnerSideB=
  "58b1a3f2-f94b-48d1-ab5e-fbc4857cdfbc:CommunicationPortInterface" />
<InternalLink Name="Link To Port_4" RefPartnerSideA=
  "58b1a3f2-f94b-48d1-ab5e-fbc4857cdfbc:CommunicationPortInterface" RefPartnerSideB=
  "65307e3e-95fd-41ac-9982-e5e4fffc2fb15:CommunicationPortInterface" />
...
</InternalElement>
</InstanceHierarchy>
</CAEXfile>
```

Attributes

Attributes are assigned to internal elements as follows:

```
...
<InternalElement ID="1d1a37ed-19d9-4a23-bc91-51f5a8e0244b" Name="Ungrouped devices">
  <InternalElement ID="ab193f5d-0375-4a6d-a576-a903e2b77cc" Name="ET 200SP station_1">
    <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
      <Value>System:Device.ET200SP</Value>
    </Attribute>
    <InternalElement ID="7636c362-a7af-47bb-8a18-e6428a6d61ff" Name="Rack_0">
      <Attribute Name="TypeName" AttributeDataType="xs:string">
        <Value>Rack</Value>
      </Attribute>
      <Attribute Name="PositionNumber" AttributeDataType="xs:int">
        <Value>0</Value>
      </Attribute>
      <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
        <Value>False</Value>
      </Attribute>
      <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
        <Value>System:Rack.ET200SP</Value>
      </Attribute>
    ...
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath=
    "AutomationProjectConfigurationRoleClassLib/Device" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath=
  "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
<InternalLink Name="Link To Port_1" RefPartnerSideA=
  "5758e2ff-3974-41e9-8bcc-b61a23f1bb58:CommunicationPortInterface"
  RefPartnerSideB="46683602-5129-4504-a9d1-48e6421e2cf0:CommunicationPortInterface" />
...
</InternalElement>
```

Handling modes for attributes

The handling of attributes is defined for every attribute individually as follows:

- Ignored
The attribute will be ignored during import and is not present in the export file.
- Mandatory
The attribute has to be present in an import file and may not be deleted in the export file.
- Optional
If this attribute is missing in the import file, the default value for the attribute is specified. This attribute is missing in an export file if it is not applicable for an object, e. g. not all modules have a FirmwareVersion.

8.6 Importing/exporting hardware data

- Export-only

The attribute value is determined by the TIA Portal internally, e. g. the type name of a device item. If it is present in an import file, it will be ignored by the TIA Portal during import.

- Import-only

The attribute can influence the import behavior. If the attribute is missing in an import file, the behavior will correlate to the standard value for the attribute.

See also

[AML type identifiers \(Page 898\)](#)

8.6.5 AML type identifiers

Internal elements

The `TypeIdentifier` string consists of several parts:

- `<TypeIdentifierType>:<Identifier>`

The following values for `TypeIdentifierType` are possible:

- `OrderNumber` used to specify physical modules
- `GSD` used to specify GSD/GSDML based devices
- `System` used to specify systems and generic devices

Type identifier type: OrderNumber

`OrderNumber` is the common type identifier for all modules present in the hardware catalog, excluding GSD. AML type identifier are not always equal to TIA Portal Openness type identifier. AML type identifier do not have a `FirmwareVersion` info. The information about firmware versions is handled in a separate AML attribute "FirmwareVersion".

The format for this `TypeIdentifierType` is as following:

- <OrderNumber>
Example: OrderNumber:3RK1 200-0CE00-0AA2

Note

Wildcards in order numbers

There are a few modules in the hardware catalog which use "wildcard" characters in their order number to represent a certain cluster of real hardware, e. g. the different lengths of S7-300 racks.

In this case the specific `OrderNumber` and the "wildcard"-`OrderNumber` can both be used to create an instance of the hardware object. However, you cannot generically use wildcards at any position. Example: An S7-300 rack can be created in the following ways:

`OrderNumber:6ES7 390-1***0-0AA0`

or

`OrderNumber:6ES7 390-1AE80-0AA0`

Regard that you cannot use the following structure for instance:

`OrderNumber:6ES7 390-1AE80-0A*0`

The return value of reading the type identifier is always the order number from the hardware catalog.

Example: Reading `OrderNumber:6ES7 390-1AE80-0AA0`
returns `OrderNumber:6ES7 390-1***0-0AA0`

Type identifier type: GSD

The type identifier for GSD and GSDML based devices is `TypeIdentifier = GSD:<Identifier>`

The identifier is composed by the following elements

- `GsdName`: name of the GSD or GSDML in uppercase letters
- `GsdType`: One of the following:
 - D: Device
 - R: Rack
 - DAP: HeadModule
 - M: Module
 - SM: Submodule
- `GsdId`: ID of the GSD or GSDML

8.6 Importing/exporting hardware data

The following formats for the type identifier are supported of the CAx import/export:

- GSD.<GsdName>/<GsdType>

Examples:

GSD:SIEM8139.GSD/DAP

GSD:GSDML-V2.31-SIEMENS-SINAMICS_DCP-20140313.XML/D

- <GsdName>/<GsdType>/<GsdId>

Examples:

GSD:SIEM8139.GSD/M/4

GSD:GSDML-V2.31-SIEMENS-SINAMICS_G110M-20140704.XML/M/IDM_DRIVE_47

Type identifier type: System

System. is the identifier for objects that cannot be determined by any other identifier. The formats for this TypeIdentifierType are as following:

- <SystemTypeIdentifier>

Examples:

System:Device.S7300

System:Subnet.Ethernet

- <SystemTypeIdentifier>/<AdditionalTypeIdentifier>

The AdditionalTypeIdentifier is necessary in case the SystemTypeIdentifier is not unique.

The SystemTypeIdentifier has a prefix for certain object types:

Subnet.

Device.

Rack.

Example: System:Rack.S71600/Large

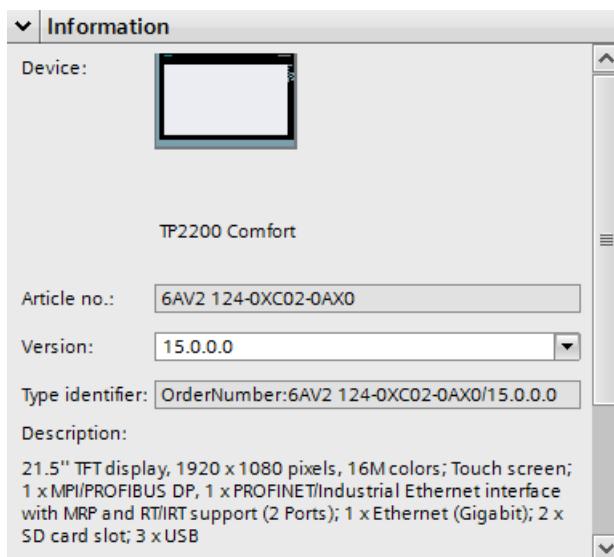
A rack with an ordner number is identified via the OrderNumber identifier.

Displaying type identifiers in TIA Portal

If you need to know a type identifier you inquire it in TIA Portal as follows:

1. Enable the setting "Enable display of the type identifier for devices and modules" in "Options > Settings > Hardware configuration > Display of the type identifier".
2. Open the editor "Devices & networks".
3. Select a device in the Catalog.

The type identifier is displayed in the viewlet "Information"



8.6.6 Export/Import of base unit Information via AML

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- PLC is offline

Application

In TIA Portal, you can export and import base unit information from and to TIA Portal so as to facilitate the information exchange with other tools like EPLAN.

There are two types of base units supported during export and import in TIA Portal project:

- Single base unit
- Double base unit

Export of base unit

For example, a module configured with base unit information in TIA Portal, CAx shall export base unit information as a 'sub-module' under a module in the AML file.

The base unit submodule shall always be exported with:

- PositionNumber: 0
- DeviceItem: Accessory
- BuiltIn: False
- TypeIdentifier: "The typeID of the Baseunit"
- ID: Always randomly generated GUID

Export of single base unit

The below example shows an AML file for DI module configured with a single base unit in TIA Portal

```
<InternalElement ID="6f76c890-5c5d-41c4-9ade-96543b0222ac" Name="DI 8x24VDC ST_1">
...
<InternalElement ID="69233c1f-7ef7-4999-8e84-691d0ff3a210" Name="BaseUnit">
<Attribute Name="DeviceItemType" AttributeDataType="xs:string">
<Value>Accessory</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>0</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6EST 193-6BP00-0DA0</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
...

```

Export of double base unit

The below example shows an AML file for two DI modules configured with double base unit in TIA Portal. For example, first module shall be configured with base unit having prefix 'IX300' and second shall be configured with same double base unit having prefix 'IX301'.

During export, only first module which is configured with base unit having IX300 suffix shall be exported with a base unit sub-module in the AML file. The second module, the one configured with IX301 shall not be injected with any Sub-module under it.

```

<InternalElement ID="6f76c890-5c5d-41c4-9ade-96543b0222ac" Name="DI 8x24VDC ST_1">
...
<InternalElement ID="3a1bee8a-12d0-4ec4-849c-333d45113d9c" Name="BaseUnit">
<Attribute Name="DeviceItemType" AttributeDataType="xs:string">
<Value>Accessory</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>0</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 193-6BP60-0DA0</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
...
<InternalElement ID="5f843491-b053-4dc8-b879-9ac327ee2a7e" Name="DI 8x24VDC ST_2">
...
<InternalElement ID="55c30280-6f8a-4c37-9b2d-41bb90941258" Name="DI 8x24VDC ST_2">
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value> </Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
...

```

Import of base unit

It shall be possible to import a module configured with a base unit sub-module in an AML file.

During import of base unit,

- GUID, PositionNumber, BuiltIn, and DeviceItem has no relevance, and hence it is not displayed in TIA Portal
- If any unintended module has base unit sub-module in AML file, Openness shall not return "BaseUnit" attribute for it. Hence, CAx shall display a proper warning for it in the log file.
- CAx shall not verify for 'correctness' of base unit MLFB in AML file (except using it to identify whether it is single or double BU). CAx shall try to set the MLFB of base unit through Openness. In case of an error from Openness a proper error will be displayed.
- Import of previous version AML file with/without any BaseUnit information should be successful by importing information into TIA Portal.

Import of single base unit

During single base unit import, only the module carrying the base unit submodule in the AML file shall be imported with base unit information in TIA Portal.

8.6 Importing/exporting hardware data

Single BaseUnit shall be identified from the TypelIdentifier in AML file with a specific pattern as below:

OrderNumber:xxxx 193-6[B|U|T]xYx-xxxx where value of Y (11th Position) can be in the range of 0 to 5.

Import of double base unit:

During double base unit import, two modules(adjacent) shall be displayed with base unit information in TIA Portal. The first module configured with double base unit submodule in AML file shall be imported with a double base unit by appending suffix '|X300' to base unit MLFB. The second module, the one not having any base unit sub-module under it shall be imported with same double base unit by appending suffix '|X301' to base unit MLFB. Double BaseUnit shall be identified from the TypelIdentifier in AML file with a specific pattern as below:

OrderNumber:xxxx 193-6[B|U|T]x6x-xxxx

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

8.6.7 Export/Import AML with extension rack connection

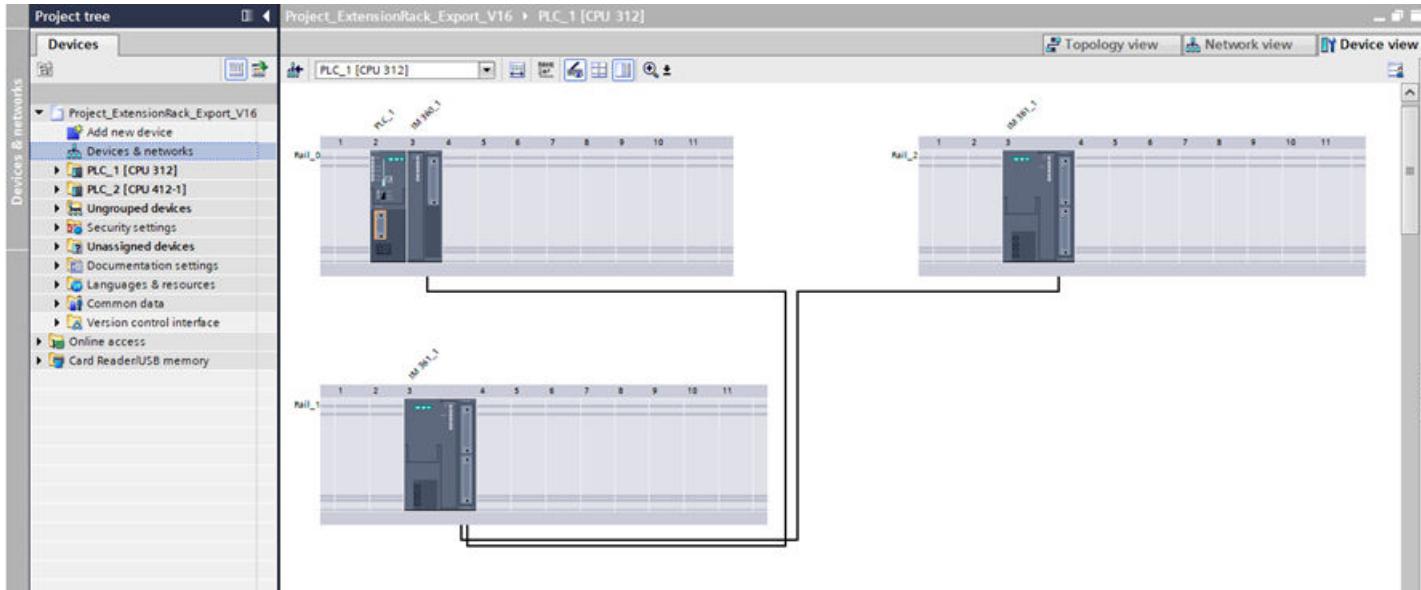
Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open
See [Opening a Project \(Page 106\)](#)

Application

In TIA Portal, you can export the devices with multiple racks having extension rack connection to an AML file and also import it back to get the same device configuration created back in the TIA Portal project.

Example: Device with multiple racks having extension rack connections



AML structure

In TIA Portal, extension rack connections between multiple racks are modeled under sender and receiver modules (both are DeviceItem objects) directly. However as per AR APC recommendation, these connections to be modeled as Port-to-Port connection under a CommunicationPort. A dummy CommunicationInterface having dummy CommunicationPort objects shall be added under IM modul to make inline with the recommendation.

8.6 Importing/exporting hardware data

The following sample shows a partial element structure of the exported AML file for above device configuration:

```
<InternalElement ID="1ddb8d5c-d6cc-42c9-b1d8-621219b139f6" Name="RackExtension">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>ExtensionRack</Value>
</Attribute>
...
<InternalElement ID="f25e531a-1793-4896-ade5-a87bd98de06e" Name="IM 46x SenderPort_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X1</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="9a824a06-89b9-4ba8-bee0-83c89b1f5e53"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
...
<InternalElement ID="d841278f-558a-41ab-9f03-91eeb454dc6b" Name="RackExtension">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>ExtensionRack</Value>
</Attribute>
...
<InternalElement ID="2e1ff3b2-8a3e-4d5b-a95c-3ed027287db9" Name="IM 46x ReceiverPort_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X1</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="98460c75-a05d-4c23-8f88-33878ccd79c5"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<InternalElement ID="7615a32e-09dc-4171-88ed-118026357bae" Name="IM 46x SenderPort_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X2</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>2</Value>
</Attribute>
```

8.6 Importing/exporting hardware data

```
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="846093a9-6473-4946-acf4-95a7813924df"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
...
<InternalElement ID="f4eb31c6-41d4-4a6e-ac33-de9c054a8c74" Name="RackExtension">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>ExtensionRack</Value>
</Attribute>
...
<InternalElement ID="c11d2227-91db-41ae-9d94-d822e3ab9c7a" Name="IM 46x ReceiverPort_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X1</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="a9b2cce1-7078-4347-b5f2-428dalad5326"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
...
<InternalLink Name="Link To Port_1" RefPartnerSideA="f25e531a-1793-4896-ade5-
a87bd98de06e:CommunicationPortInterface" RefPartnerSideB="2e1ff3b2-8a3e-4d5b-
a95c-3ed027287db9:CommunicationPortInterface" />
<InternalLink Name="Link To Port_2"
RefPartnerSideA="7615a32e-09dc-4171-88ed-118026357bae:CommunicationPortInterface"
RefPartnerSideB="c11d2227-91db-41ae-9d94-d822e3ab9c7a:CommunicationPortInterface" />
```

Note

The extension rack interface shall be exported only when extension rack connections are present. Also, the number of dummy ports added under ExtensionRack dummy interface is based on ports participating in extension rack connection at module level. In above example, the "IM 460-0_1" module supports two ports (IM 46x SenderPort_1 and IM 46x SenderPort_2). However, only one port is configured with connection in TIA Portal. Hence, the exported AML file shall contain only one port under extension rack interface.

Extension rack connection

The XML representation of extension rack connections between multiple racks shall be done using format shown below.

ExternalInterface-

<ExternalInterface> internal element shall be added under <CommunicationPort> internal element which participates in the connection

```

<InternalElement ID="[IM Module Unique ID]" Name="[IM Module Name]">
...
<InternalElement ID="[Dummy Interface Unique ID]" Name="RackExtension">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>ExtensionRack</Value>
</Attribute>
...
<InternalElement ID="[Dummy Port Unique ID]" Name="[IM Module Sender/Receiver Name]">
...
<ExternalInterface ID="[External Interface Unique ID]" Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<InternalElement ID="[Dummy Port Unique ID]" Name="[IM Module Sender/Receiver Name]">
...
<ExternalInterface ID="[External Interface Unique ID]" Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
```

8.6 Importing/exporting hardware data

Internal link

Extension racks connections are represented using `<InternalLink>` tags. The `<InternalLink>` tags shall be added under common parent of multiple racks (i.e., Device). Internal link name shall be unique across the common parent.

```
<InternalLink Name="Link To [Internal link Name]" RefPartnerSideA="[Communication Port UniqueID]:[Communication Port External Interface Name]" RefPartnerSideB="[Communication Port UniqueID]:[Communication Port External Interface Name]" />
```

8.6.8 Connection handling for extension racks

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)

Application

You can use the TIA Portal Openness to fetch, add, and remove extension rack connections so that you can make use of TIA Portal Openness to implement extension rack connection support during CAx export/import.

Program code

```
ImConnection imConnection = portDeviceItem.GetService<ImConnection>();  
imConnection.Connect(partnerport);  
imConnection.Disconnect();  
imConnection.GetPartnerPort();  
var imConnectionOwner = imConnection.OwnedBy;
```

See also

- Connecting to the TIA Portal (Page 76)
Opening a project (Page 106)

8.6.9 Export of CAx data

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

In TIA Portal you can export your configuration in the device&networks editor to an AML file. This function is based on TIA Portal Openness and enables you to export hardware data from project or device level.

TIA Portal Openness provides the following ways to export CAx data:

- Export function
The export function is accessed via `CaxProvider` service. To get the `CaxProvider` service invoke the `GetService` method at `Project` object.
- Command line interface
You execute the "Siemens.Automation.Cax.AmiHost.exe" located in "C:\Program Files \Siemens\Automation\Portal V..\\Bin\" by passing specific command line arguments.

Export and Import restrictions for CAx

CAx does not support the export and import of

- Port-Port connections
- Connections to and between extension racks
- Multi-CPUs
- HMI devices except push button panels and key panels
- Drives
- Output mode and range of analog channels
- Packed addresses

CAx does not support the export and import of the following device and drives:

- 6AV2 104-0XXXX-XXXX
- 6AV2 155-0XXXX-XXXX
- 6ES7 XXX-XXXXX-XXXX
- 6ES7 370-0AA01-0AA0
- 6ES7 451-3AL00-0AE0
- 6GK5 414-3FC00-2AA2
- 6GK5 414-3FC10-2AA2

8.6 Importing/exporting hardware data

- 6GK5 495-8BA00-8AA2
- 6GK5 496-4MA00-8AA2
- 6GK5 602-0BA00-2AA3
- 6GK5 602-0BA10-2AA3
- 6GK5 612-0BA00-2AA3
- 6GK5 612-0BA10-2AA3
- 6GK5 613-0BA00-2AA3
- 6GK5 623-0BA10-2AA3
- 6GK5 627-2BA10-2AA3
- System:Device.Scalance/S627
- System:IPProxy.Device
- System:IPProxy.Rack

Program code: Access the CaxProvider service

Modify the following program code to access the CaxProvider service:

```
Project project = tiaPortal.Projects.Open(...);
CaxProvider caxProvider = project.GetService<CaxProvider>();

if(caxProvider != null)

{
    // Perform CAx export and import operation
}
```

CAx export at project level

To export CAx data at project level, use the `Export` method with the following parameters:

Name	Example	Description
ProjectToExport	tiaPortal.Projects[0]	Project object to Export
ExportFilePath	new FileInfo(@"D:\Temp\ProjectExport.aml")	Full Export file path of AML file
LogFilePath	new FileInfo(@"D:\Temp\ProjectExport_Log.log")	Full file path of Log file

Modify the following program code to export CAx data at project level:

```
caxProvider.Export(project, new FileInfo(@"D:\Temp\ProjectExport.aml"),
new FileInfo(@"D:\Temp\ProjectExport_Log.log"));
```

CAx export at device level

To export CAx data at device level, use the `Export` method with the following parameters:

Name	Example	Description
DeviceToExport	<code>project.Devices[0]</code>	Device object to Export
ExportFilePath	<code>new FileInfo(@"D:\Temp\ProjectExport.aml")</code>	Full Export file path of AML file
LogFilePath	<code>new FileInfo(@"D:\Temp\ProjectExport_Log.log")</code>	Full file path of Log file

Modify the following program code to export CAx data at project level:

```
caxProvider.Export(device, new FileInfo(@"D:\Temp\DeviceExport.aml"), new
FileInfo(@"D:\Temp\DeviceExport_Log.log"));
```

CAx export via command line

To export CAx data via command line, use "Siemens.Automation.Cax.AmiHost.exe" with the following parameters:

Parameter	Example	Description
-p	-p "D:\Temp\MyProject.apx"	Specifies a full path name to an existing TIA Portal project.
-d	-d "S7300/ET200M station_1"	Optional parameter. If no device is specified export will take place at project level. Specifies the name of the device or station inside the specified TIA project, that needs to be exported.
-m	-m "AML"	Specifies the export/import mode (format for export/import): "AML" exports in AML format
-e	-e "D:\Import" -e "D:\Import\CAx_Export.aml"	Specifies full path of AML file to be exported. The project name will be used as exported file name if only a path is specified.

Modify the following program code to export CAx data at project level via the command line:

```
//please adapt the path and the extension apx to the installed version of
TIA Portal
Siemens.Automation.Cax.AmiHost.exe -p "D:\Temp\MyProject.apx" -m "AML" -e
"D:\Import\CAx_Export.aml"
```

8.6 Importing/exporting hardware data

Modify the following program code to export CAx data at device level via the command line:

```
//please adapt the path and the extension apx to the installed version of  
TIA Portal  
Siemens.Automation.Cax.AmiHost.exe -p "D:\Temp\MyProject.apx" -d "S7300/  
ET200M station_1" -m "AML" -e "D:\Import\CAx_Export.aml"
```

8.6.10 Import of CAx data

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)

Application

In TIA Portal you can import your configuration in the device&networks editor from an AML file. This function enables you to import hardware data from project or device level.

TIA Portal Openness provides the following ways to export CAx data:

- Import function
The import function is accessed via `CaxProvider` service. To get the `CaxProvider` service invoke the `GetService` method at `Project` object.
- Command line
You execute the "Siemens.Automation.Cax.AmiHost.exe" located in "C:\Program Files \Siemens\Automation\Portal V..\\Bin\" by passing specific command line arguments:

Program code: Access the CaxProvider service

Modify the following program code:

```
//Access the CaxProvider service  
Project project = tiaPortal.Projects.Open(...);  
CaxProvider caxProvider = project.GetService<CaxProvider>();  
  
if(caxProvider != null)  
{  
    // Perform Cax export and import operation  
}
```

CAx import

To import CAx data into a TIA Portal project, you use the `Import` method with the following parameters:

Name	Example	Description
ImportFilePath	<code>new FileInfo(@"D:\Temp\ProjectExport.aml")</code>	Full import file path of AML file
LogFilePath	<code>new FileInfo(@"D:\Temp\ProjectExport_Log.log")</code>	Full file path of log file
ImportOptions	<code>CaxImportOptions.MoveToParkingLot</code> <code>CaxImportOptions.RetainTiaDevice</code> <code>CaxImportOptions.OverwriteTiaDevice</code>	Conflict resolution strategies in case of importing into an already existing non empty project.

Modify the following program code to import CAx data:

```
caxProvider.Import(new FileInfo(@"D:\Temp\ProjectImport.aml"), new
FileInfo(@"D:\Temp\ProjectImport_Log.log"),
CaxImportOptions.MoveToParkingLot);
```

The following `CaxImportOptions` are provided:

Import option	Description
<code>MoveToParkingLot</code>	Retain name conflicting device/s in the project and import those out of CAx into a parkinglot folder
<code>RetainTiaDevice</code>	Retain name conflicting device/s in the project and do not import those out of CAx
<code>OverwriteTiaDevice</code>	Overwrite name conflicting device/s in the project by the ones out of CAx

CAx import via command line

To import CAx data via command line, use "Siemens.Automation.Cax.AmiHost.exe" with the following parameters:

Parameter	Example	Description
<code>-p</code>	<code>-p "D:\Temp\MyProject.ap"</code>	Specifies a full path name to an existing TIA Portal project.
<code>-m</code>	<code>-m "AML"</code>	Specifies the export/import mode (format for export/import): "AML" exports in AML format
<code>-i</code>	<code>-i "D:\Import\CAx_Export.aml"</code>	Specifies full path of the AML file to be imported.
<code>-c</code>	<code>-c "ParkingLot"</code>	Specifies different strategies if there is a conflict in the device name according to import options.

8.6 Importing/exporting hardware data

Modify the following program code to import CAx data via the command line:

```
//please adapt the path and the extension apx to the installed version of  
TIA Portal  
Siemens.Automation.Cax.AmiHost.exe -p "D:\Temp\MyProject.apx" -m "AML" -i  
"D:\Import\CAx_Export.aml"
```

The following import options are provided:

Import option	Description
ParkingLot	Retain name conflicting device/s in the project and import those out of CAx into a parkinglot folder
RetainTia	Retain name conflicting device/s in the project and do not import those out of CAx
OverwriteTia	Overwrite name conflicting device/s in the project by the ones out of CAx

8.6.11 Export/Import of sub modules

Requirements

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- PLC is offline

Application

You can have round trip exchange of sub modules data between the TIA Portal and other engineering tools, e.g. CAD tool like EPLAN by keeping a common hierarchy for sub modules inside AML file during export and import. For example, the sub modules like Bus Adapters shall have different internal hierarchy in TIA Portal than in other applications (e.g., CAD tools like EPLAN).

AML structure of the export file

You can export sub modules data from TIA Portal hierarchy to AML file hierarchy.

The following example depicts the partial AML file structure that shall be generated during the export of Bus Adapter as sub module from TIA Portal .

8.6 Importing/exporting hardware data

```
<?xml version="1.0" encoding="utf-8"?>
<CAEXfile FileName="Project4.aml" SchemaVersion="2.15"
xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd">
<AdditionalInformation>
<WriterHeader>
<WriterName>Totally Integrated Automation Portal</WriterName>
<WriterID>1d4fcebb-1ad6-4881-b01d-bca335d94a46:V1.0.</WriterID>
<WriterVendor>Siemens AG</WriterVendor>
<WriterVendorURL>www.siemens.com</WriterVendorURL>
<WriterVersion>15</WriterVersion>
<WriterRelease>1500.0100.0.0</WriterRelease>
<LastWritingDateTime>2018-05-03T11:23:10.3011329Z</LastWritingDateTime>
</WriterHeader>
</AdditionalInformation>
<AdditionalInformation AutomationMLVersion="2.0" />
<AdditionalInformation DocumentVersions="Recommendations">
<Document DocumentIdentifier="AR APC" Version="1.1.0" />
</AdditionalInformation>
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
<InternalElement ID="6cd7f80f-e049-4958-ba67-630481805bf0" Name="Project4">
<Attribute Name="ProjectManufacturer" AttributeDataType="xs:string" />
<Attribute Name="ProjectSign" AttributeDataType="xs:string" />
<Attribute Name="ProjectRevision" AttributeDataType="xs:string" />
<Attribute Name="ProjectInformation" AttributeDataType="xs:string" />
<InternalElement ID="b27045c4-9cb3-4b8d-916b-85f8100d1602" Name="Ungrouped devices">
<InternalElement ID="3f770698-940d-49c2-9f77-06fc458e1340" Name="ET 200SP station_1">
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>System:Device.ET200SP</Value>
</Attribute>
<InternalElement ID="6f52fbab-a221-4d54-9368-84c392ca7fec" Name="Rack_0">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>Rack</Value>
...
<InternalElement ID="f7445c0b-1c52-4a84-915f-2c8bee13af70" Name="BA 2xRJ45">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>BA 2xRJ45</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>127</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6EST 193-6AR00-0AA0</Value>
</Attribute>
<Attribute Name="FirmwareVersion" AttributeDataType="xs:string">
<Value>V0.0</Value>
</Attribute>
<InternalElement ID="40f8bbce-35d3-4d65-907a-bece3e0144e0" Name="PROFINET interface">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X1</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
```

```

<Value>1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<InternalElement ID="8fb775eb-96c6-48d6-af8a-96ba72418830" Name="IE1">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>Ethernet</Value>
</Attribute>
<Attribute Name="NetworkAddress" AttributeDataType="xs:string">
<Value>192.168.0.1</Value>
</Attribute>
<Attribute Name="SubnetMask" AttributeDataType="xs:string">
<Value>255.255.255.0</Value>
</Attribute>
<Attribute Name="IpProtocolSelection" AttributeDataType="xs:string">
<Value>Project</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Node" />
</InternalElement>
<InternalElement ID="f28a3d93-d821-4556-9df1-a45f0e4ff6a6" Name="Port_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>P1 R</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<InternalElement ID="ad6a0faa-3b70-4528-8c54-8183018b6714" Name="Port_2">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>P2 R</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>2</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
...

```

Note

CAx shall perform export and import of AR APC version in AML file in accordance to the installed version of TIA Portal.

Import sub modules

You can import sub modules from an AML files, which is generated from above export.

Note

- The export hierarchy change behavior shall be applicable only in versions V15.1 onwards
- The hierarchy in AML file shall not influence/affect the TIA Portal internal hierarchy after Import.
- The AML files created using older TIA Portal versions shall also be imported without any failure.
- This hierarchy change/transformation behavior is applicable for both built-in and non built-in sub modules.

Multiple sub modules under same Interface

There are some scenarios where multiple sub modules shall exist under a same interface. For ex: IO Device : IM 155-6 PN/3 HF 6ES7 155-6AU30-0CN0/V4.2. This head module has two non built-in Bus Adapters under a same interface. In such case, it shall be possible to export mentioned Bus Adapters from TIA Portal hierarchy to required AML file hierarchy. In this example from TIA Portal hierarchy, 'PROFINET interface' has two Bus Adapters, three ports and one node. Here, Port_1 and Port_2 logically belongs to BA 2xRJ45 and Port_3 logically belongs to BA 2xRJ45_1 though all the three ports are aggregated under one interface.

During Export:

- Only first sub module shall get 'original' interface along with its connection relevant information. Here, BA 2xRJ45 gets original interface along with node 'IE1', 'Port_1' and 'Port_2'.
- Rest of the sub modules shall get a 'duplicate' interface with ports which logically belongs to the sub module. Here, BA 2xRJ45_1 shall get a 'duplicate' interface and Port_3.
- If the head module is connected to a subnet/loSystem, the relevant link information(like ExternalInterface links) shall be exported only as part of first sub module (ExternalInterface link related to Subnet under 'Node' and ExternalInterface link related to loSystem under 'Interface') .
- The link information pertaining to topology connection, shall be part of respective 'Port'.

During import:

- It shall be possible to import multiple sub modules from an AML file which is generated out of above mentioned export.
- It shall be possible that EPLAN generated AML file shall have node information inside secondary interface

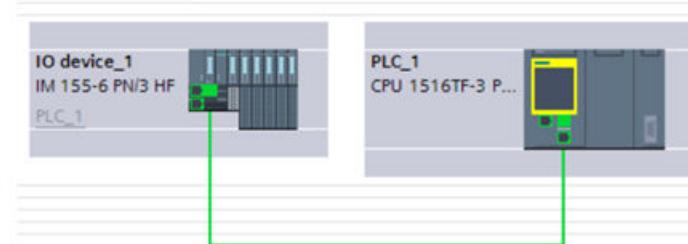
- Node details, which is duplicate of primary interface node are handled as mentioned below:
 - Node Attributes: Overwrites node attribute details which set during Primary interface processing
 - Subnet Connection: Silently skipped if it is already connected otherwise connection shall be established
- If the AML file contains IoSystem connection details on secondary interface, then
 - Connection shall be skipped if already connected and user shall be notified with proper error message in InfoTab.
 - Connection shall be established if it is not connected.

The following configuration shows IO device configuration with master-slave and topology connections.

Network View



Topology View



8.6 Importing/exporting hardware data

The below example shows a partial AML file that shall be generated during export for the above configuration:

```

<?xml version="1.0" encoding="utf-8"?>
<CAEXFile FileName="MultipleBA_01.aml" SchemaVersion="2.15"
xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd">
<AdditionalInformation>
<WriterHeader>
<WriterName>Totally Integrated Automation Portal</WriterName>
<WriterID>1d4fccebb-1ad6-4881-b01d-bca335d94a46:V1.0</WriterID>
<WriterVendor>Siemens AG</WriterVendor>
<WriterVendorURL>www.siemens.com</WriterVendorURL>
<WriterVersion>15</WriterVersion>
<WriterRelease>1501.0000.0.0</WriterRelease>
<LastWritingDateTime>2018-05-17T09:36:46.9230179Z</LastWritingDateTime>
</WriterHeader>
</AdditionalInformation>
<AdditionalInformation AutomationMLVersion="2.0" />
<AdditionalInformation DocumentVersions="Recommendations">
<Document DocumentIdentifier="AR APC" Version="1.1.0" />
</AdditionalInformation>
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
<InternalElement ID="e005c094-1b0a-42c4-92a0-67c981508c1a" Name="Project45">
<Attribute Name="ProjectManufacturer" AttributeDataType="xs:string" />
<Attribute Name="ProjectSign" AttributeDataType="xs:string" />
<Attribute Name="ProjectRevision" AttributeDataType="xs:string" />
<Attribute Name="ProjectInformation" AttributeDataType="xs:string" />
<InternalElement ID="2782e61d-8c27-46cb-93ea-6b804157ae60" Name="PN/IE_1">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>Ethernet</Value>
</Attribute>
<ExternalInterface ID="2d901881-a2bf-4fe7-915f-b2542b346988" Name="LogicalEndPoint_Subnet">
RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
Subnet" />
...
<InternalElement ID="dc5cf410-2516-4b0b-ad1a-c43117d8c9b3" Name="BA 2xRJ45">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>BA 2xRJ45</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>127</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 193-6AR00-0AA0</Value>
</Attribute>
<Attribute Name="FirmwareVersion" AttributeDataType="xs:string">
<Value>V0.0</Value>
</Attribute>
<InternalElement ID="f04874a8-2d35-47c4-93ae-d6fdc2668479" Name="PROFINET interface">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X1</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>

```

8.6 Importing/exporting hardware data

```
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="81a7d9df-99b8-4eca-8e72-404b22bd05e7"
Name="LogicalEndPoint_Interface" RefBaseClassPath="CommunicationInterfaceClassLib/
LogicalEndPoint" />
<InternalElement ID="83eb7d69-8cd5-4217-a07a-0c656d215ec7" Name="IE1">
...
...
```

Pruned AML

Pruning is an act of optimizing the content by removing certain things which are not necessary to be provided. For information on Pruned AML, (Refer Pruned AML (Page 890)).

There might be some scenarios where sub module configuration hierarchy is not same in TIA Portal and CAD tools (like EPALN) due to pruned sub modules. In such scenarios, TIA Portal shall support import of both pruned and unpruned AML files.

Note

- TIA Portal shall always export unpruned AML file.
 - TIA Portal shall always import both pruned and unpruned AML file
-

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

8.6.12 Import CAx data without logical address

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open
See [Opening a project \(Page 106\)](#)

Application

In TIA Portal, you can use CAx import to configure connection between channel and tag without having the start address of an I/O module and/or logical address of tag specified in the AML file.

The following AML file example depicts the XML that shall be generated without start address and logical address attributes.

8.6 Importing/exporting hardware data

```
<?xml version="1.0" encoding="utf-8"?><CAEXFile FileName="TagsExport.aml"
SchemaVersion="2.15" xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd">
...
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
<InternalElement ID="fff25423-fe9e-4334-9331-4cec118e06f7" Name="Project1">
...
<InternalElement ID="59c0b48b-aa6c-45c3-8dc8-bba5367bd4fb" Name="S7300/ET200M station_1">
...
<InternalElement ID="1b7b2b24-243b-4348-831e-bc46bc35957f" Name="Rail_0">
...
<InternalElement ID="974ca791-ad8d-482b-be80-2cf4e8dcedaf" Name="PLC_1">
...
<InternalElement ID="9564bcc2-8ea0-4be7-a950-5c55b34e474a" Name="Default tag table">
<ExternalInterface ID="7fd969e6-c2c9-45a8-b573-68833df327f5" Name="Tag_1"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Tag">
<Attribute Name="DataType" Attribute DataType="xs:string">
<Value>Bool</Value>
</Attribute>
</ExternalInterface>
<ExternalInterface ID="33899862-86c1-4171-832a-1136b6e59b9d" Name="Tag_2"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Tag">
<Attribute Name="DataType" Attribute DataType="xs:string">
<Value>Byte</Value>
</Attribute>
</ExternalInterface>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
TagTable" />
</InternalElement>
...
<Attribute Name="PositionNumber" Attribute DataType="xs:int">
<Value>8</Value>
</Attribute>
<Attribute Name="BuiltIn" Attribute DataType="xs:boolean">
<Value>true</Value>
</Attribute>
<Attribute Name="Address">
<RefSemantic CorrespondingAttributeName="OrderedListType" />
<Attribute Name="1">
<Attribute Name="StartAddress" Attribute DataType="xs:int">
<Value>832</Value>
</Attribute>
<Attribute Name="Length" Attribute DataType="xs:int">
<Value>128</Value>
</Attribute>
<Attribute Name="IoType" Attribute DataType="xs:string">
<Value>Input</Value>
</Attribute>
</Attribute>
<Attribute Name="2">
<Attribute Name="StartAddress" Attribute DataType="xs:int">
<Value>832</Value>
</Attribute>
<Attribute Name="Length" Attribute DataType="xs:int">
<Value>128</Value>
</Attribute>
```

```
<Attribute Name="IoType" AttributeDataType="xs:string">
<Value>Output</Value>
</Attribute>
</Attribute>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<InternalElement ID="29e0bb63-0050-46e3-968a-fcecf4eb050a" Name="DI 16x24VDC_1">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>DI16 x 24VDC</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>4</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 321-1BH02-0AA0</Value>
</Attribute>
<Attribute Name="Address">
<RefSemantic CorrespondingAttributePath="OrderedListType" />
<Attribute Name="1">
<Attribute Name="Length" AttributeDataType="xs:int">
<Value>16</Value>
</Attribute>
<Attribute Name="IoType" AttributeDataType="xs:string">
<Value>Input</Value>
</Attribute>
</Attribute>
</Attribute>
<ExternalInterface ID="175dc9c9-f9a3-4b10-b43e-68dfc14811fc" Name="Channel_DI_0"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Channel">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>Digital</Value>
</Attribute>
<Attribute Name="IoType" AttributeDataType="xs:string">
<Value>Input</Value>
</Attribute>
<Attribute Name="Number" AttributeDataType="xs:int">
<Value>0</Value>
</Attribute>
<Attribute Name="Length" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
</ExternalInterface>
<ExternalInterface ID="23e99053-906c-4548-9bd2-e975cacf01b2" Name="Channel_DI_1"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Channel">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>Digital</Value>
</Attribute>
<Attribute Name="IoType" AttributeDataType="xs:string">
```

8.6 Importing/exporting hardware data

```

<Value>Input</Value>
</Attribute>
<Attribute Name="Number" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="Length" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
</ExternalInterface>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
<InternalLink Name="Link To Tag_1" RefPartnerSideA="29e0bb63-0050-46e3-968a-
fcecf4eb050a:Channel_DI_0" RefPartnerSideB="9564bcc2-8ea0-4be7-a950-5c55b34e474a:Tag_1" />
<InternalLink Name="Link To Tag_2" RefPartnerSideA="29e0bb63-0050-46e3-968a-
fcecf4eb050a:Channel_DI_0" RefPartnerSideB="9564bcc2-8ea0-4be7-a950-5c55b34e474a:Tag_2" />
<InternalLink Name="Link To Tag_3" RefPartnerSideA="29e0bb63-0050-46e3-968a-
fcecf4eb050a:Channel_DI_1" RefPartnerSideB="9564bcc2-8ea0-4be7-a950-5c55b34e474a:Tag_2" />
<InternalLink Name="Link To Tag_4" RefPartnerSideA="29e0bb63-0050-46e3-968a-
fcecf4eb050a:Channel_DI_2" RefPartnerSideB="9564bcc2-8ea0-4be7-a950-5c55b34e474a:Tag_2" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
Device" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
AutomationProject" />
</InternalElement>
</InstanceHierarchy>
</CAEXfile>

```

Tag with Bool datatype (%I0.0)

For the above AML file example, upon import the Logical Address for tag is calculated by using the following algorithm:

$$\text{Logical Address} = \text{ChannelType} + \text{ByteAddress} + \text{BitAddress}$$

Name	Description
ChannelType	Input (I) or Output (Q)
ByteAddress	ByteAddress shall be calculated as specified below : StartAddress of the I/O Module * 8 + BitOffsetAddress of the I/O Module + (ChannelNumber * ChannelLength) / 8
BitAddress	BitAddress shall be calculated as specified below: (BitOffsetAddress of the I/O Module + (ChannelNumber * ChannelLength)) % 8

Note

- In cases where a tag spans across more than one modules: The algorithm above gives multiple byte addresses based on the start address of modules and an array of bit addresses per byte address, corresponding to each channel number. Then the algorithm shall select the lowest byte address and the lowest among the bit address array corresponding to this byte, for calculating the logical address of the tag. Once the logical address is assigned to the tag, the spanning is automatically taken care by the TIA Portal during import
- In TIA Portal, only few device configuration (for example ASI modules) supports Bit Offset address attribute. For modules which does not support BitOffset address attribute, default value "0" will be considered for above mentioned calculation.

Following are the list of tag data types which supports Bit address in TIA Portal:

Tag data types	Bit address value
Bool	0 to 7
LReal	Highest and lowest bit number values is 0 in TIA Portal.
LWord	
LInt	
ULInt	
LTime	
LDT	
LTime_Of_Day	

During CAx import, If a boolean tag is configured to a channel type, then logical address calculation includes bit offset address. The logical address along with bit offset shall be updated as the tag's logical address in TIA Portal UI.

Tag with other datatypes which supports Bit Offset Address

In case of tag channel configured between two different datatypes where a channel of type bool is mapped to a tag of type LDT which spans across multiple channels, then logical address calculation includes bit offset address for calculating logical address for Tag of "LDT" datatype and the tag logical address will be updated in TIA Portal UI with an error if the bit address value is other than "0" and tag channel configuration shall not happen.

You must make sure that tag-channel configuration is happening between similar datatypes.

Tag with other datatypes (%IB0)

Logical Address = ChannelType + TagDataType + ByteAddress

Name	Description
ChannelType	Input (I) or Output (Q)
TagDataType	TagDataType is an abbreviation of the tag type. Example: W for word and B for Byte
ByteAddress	ByteAddress shall be calculated as specified below : StartAddress of the I/O Module + (ChannelNumber * ChannelLength) / 8

The algorithm explained above is used to calculate the Logical Address of a tag accurately in case of

- Length of the datatype specified in the tag should be equal to the length of the channel it is mapped to.
- For example, If a tag of datatype "Byte" is mapped to an Analog channel which is of length 2Bytes: upon import of AML file which doesnot have Logical Address of the tag specified; The tag in TIA portal shall always be mapped to the first byte of the channel irrespective of which byte it was originally mapped to.

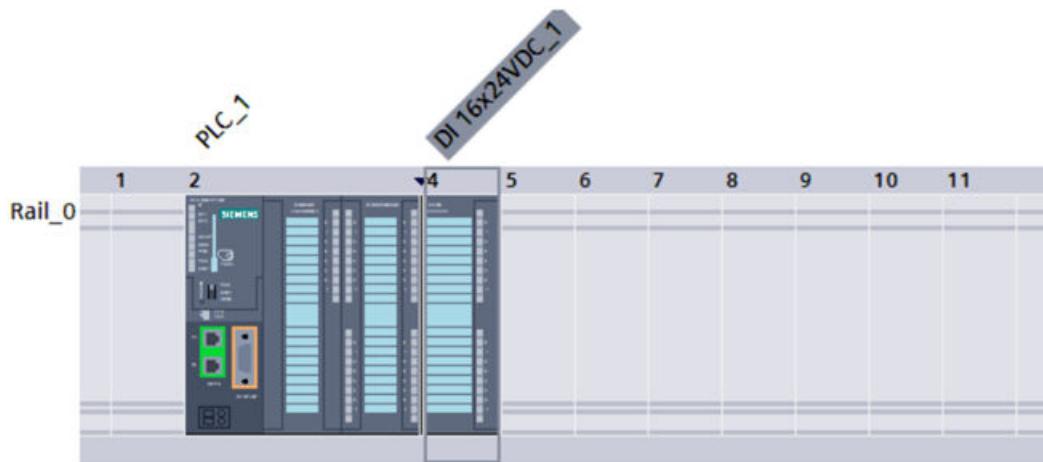
Note

- In cases where a tag spans across more than one modules: The algorithm above gives multiple byte addresses based on the Start Address of the modules. Then the algorithm shall select the lowest byte address for calculating the logical address of the tag. Once the logical address having the lowest byte address is assigned to the tag, the spanning is automatically taken care by the portal during import. If the StartAddress attribute for the I/O Module is not provided in the AML file, the default value is assigned by TIA portal and same shall be used for the above calculation.
- If the StartAddress attribute for the I/O Module is not provided in the AML file, the default value is assigned by TIA portal and same shall be used for the above calculation.

Upon successful completion of import the following tag configuration shall be created in TIA Portal for the example above.

Default tag table						
	Name	Data type	Address	Retain	Access...	Visible...
1	Tag_1	Bool	%I0.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
2	Tag_2	Byte	%IB0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
3	<Add new>				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

The channels shall be configured to the tags as depicted below.



See also

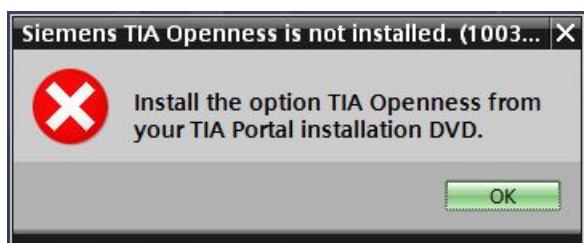
- [Connecting to the TIA Portal \(Page 76\)](#)
- [Opening a project \(Page 106\)](#)

8.6.13 Exceptions during import and export of CAx data

Exception due to non-availability of TIA Openness

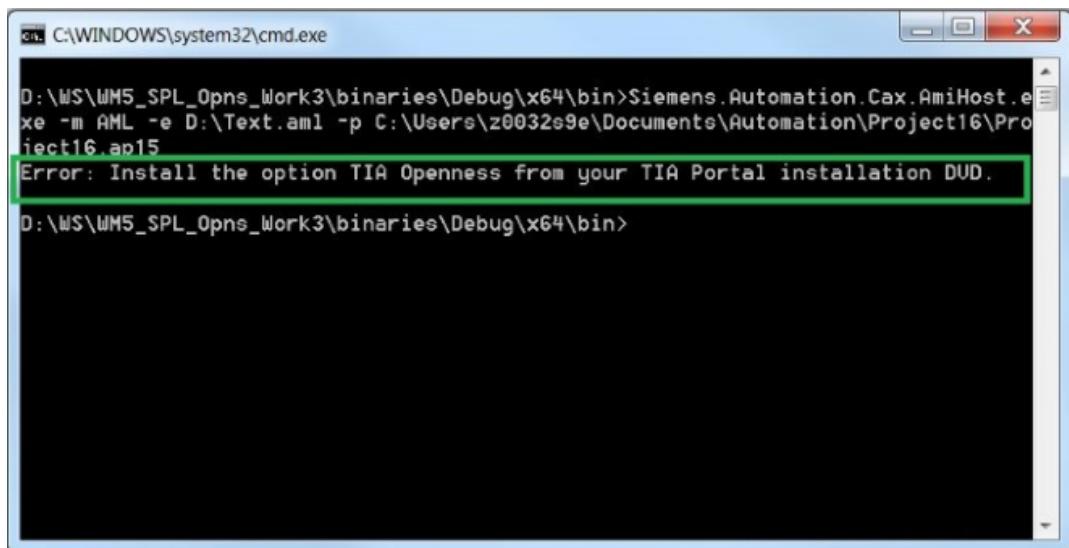
CAx implementation is based on TIA Openness Public API's. Openness Public API's are available only when user has installed Openness optional pack during TIA Portal installation. Hence, there is a need to check whether Openness is available before performing any CAx related functionalities. (Refere [Installing TIA Portal Openness \(Page 30\)](#))

Whenever user triggers a CAx Export or CAx Import actions from TIA Portal UI, a check is performed to see availability of TIA Openness in the system. If TIA Openness is not found to be installed, user will be displayed a TIA Portal message dialog with following error message dialog.



While performing the CAx operation through command-line, the following error diaglog displays during the non-availablity of TIA Openness.

8.6 Importing/exporting hardware data



A screenshot of a Windows Command Line window titled 'C:\WINDOWS\system32\cmd.exe'. The window contains the following text:

```
D:\WS\WM5_SPL_Opns_Work3\binaries\Debug\x64\bin>Siemens.Automation.Cax.AmiHost.e
xe -m AML -e D:\Text.aml -p C:\Users\z0032e9e\Documents\Automation\Project16\Pro
ject16.ap15
Error: Install the option TIA Openness from your TIA Portal installation DUD.

D:\WS\WM5_SPL_Opns_Work3\binaries\Debug\x64\bin>
```

The error message 'Error: Install the option TIA Openness from your TIA Portal installation DUD.' is highlighted with a green border.

Figure 8-3 OpennessNotInstalled-Commandline

8.6.14 Round trip exchange of devices and modules

Requirement

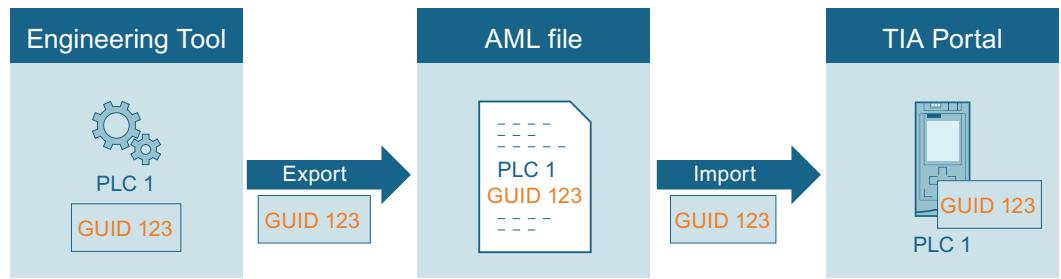
- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- PLC is offline.

Application

You can exchange configuration data between the TIA Portal and other engineering tools, e. g. an electrical planning tool like EPLAN or the TIA selection tool. For the identification of the imported and exported devices, a global unique identifier, the AML GUID, is used.

During the round trips, the AML GUID is kept stable for physical assets like devices and device items which are not built-in e.g. CPUs or modules, but not for virtual assets like tags, channels, ...

During the first export from the TIA Portal, the AML GUID for a device or a no built-in device item is randomly generated, but kept stable afterwards.

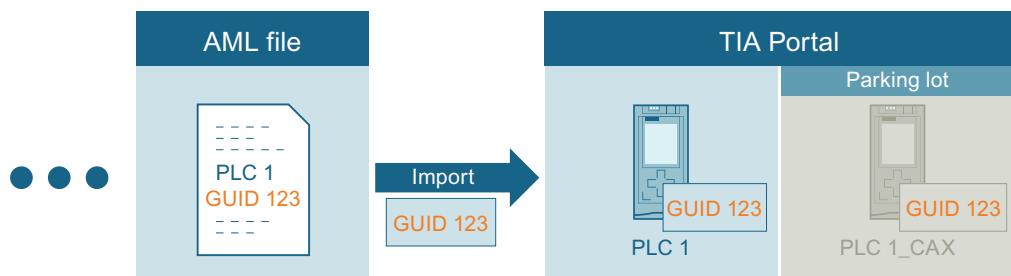


If you export a device from an engineering tool into an empty TIA Portal project, the AML GUID is added to the comment of the hardware object. If in the TIA Portal at "Tools > Settings > CAx > Import settings" the corresponding setting is enabled the AML GUID is added in the current editing language. The round trip process supports only one editing language to store the AML GUIDs. When importing or exporting data, always use the editing language with which you started the round trip.

For all following imports or exports, the AML GUID stays the same for this hardware object. Changes to the hardware object are resumed.

Within a TIA Portal project object names have to be unique. The import of a device or a device item into a TIA Portal project where a certain object with the same name already exists would lead to a naming conflict. During the import you have the possibility to move the objects with naming conflicts to the user defined parking lot. The name of the imported Object will be extended with "_CAX".

During CAx export of the GUI

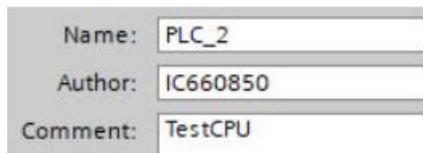


To support AML roundtrip for previous project version, during project upgrade AML GUIDs are now stored in "CustomIdentity" (App ID) instead of "Comment" part of device/deviceitem. The AML GUIDs of device/deviceitem are expected to be unique and could be stored in any editing language. So during project upgrade, all the comment editing languages are considered to get the unique AML GUID. As the AML GUID of device/deviceitem is unique, but when a new GUID with matching [AR_APPC:ID:] regex is added to comment in any editing language then the first GUID which is picked from the editing languages is considered as the AML GUID for that device/deviceitem.

Once the AML GUID's in the comment are moved to CustomIdentity, the device/deviceitem comment is updated by removing the [AR_APPC:ID:] as shown in the image below

If the comment contains multiple [AR_APPC:ID:] section then the first GUID which matches the pattern is set to CustomIdentity repository and the same is removed from comment. Rest of the text is considered as comments.

8.6 Importing/exporting hardware data



Note

Copying an imported device

If you copy a device or a device item possessing an AML GUID you have to delete the AML GUID in the comment of the copied object. Otherwise, devices or device items with identical AML GUID exists in your project and lead to an invalid AML file.

Import settings

1. Define the parking lot folder name under "Options > Settings > CAx > Settings for conflict resolution".
The parking lot folder is used to store objects with naming conflicts.
2. Activate "Options > Settings > CAx > Import settings > Save GUIDs during import".

During Import:

- The GUIDs of a physical asset shall be stored as part of its CustomIdentity section
- While importing an AML file, the GUIDs are stored as part of CustomIdentity

Note

Valid AML GUID

If you edit an AML GUID before the import, the AML GUID becomes invalid and then CAx import operation shall be aborted and corresponding information shall be logged.

Note

Exceeding length of a comment

If the appending of the AML GUID the comment exceeds its maximum limit of 500 characters, the user comment value will be trimmed to 500 characters. A corresponding information will be logged.

Export

During export:

- GUID to be exported shall be fetched from CustomIdentity for the key AR_APPC:ID.
- If the GUID is not available as part of CustomIdentity then the Multi user GUID shall be fetched and exported as the AML GUID
- If above 2 steps fail to provide the GUID of a physical asset then a new random GUID shall be generated by the export process and shall be treated as the AML GUID for physical assets.

AML structure

The generated ID is exported to AML file as depicted in the following code snippet:

```
<InternalElement ID="23aeef0-05-4116-a644-e33d43901eaf"  
Name="PLC_1"
```

8.6.15 Import AML ignoring unknown artifacts

Requirement

- The TIA Portal Openness application is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- The PLC is offline

Application

It shall be possible to import an AML file containing unknown artifacts into TIA Portal where the unknown artifacts are ignored and only relevant information consumed. Unknown artifacts contains un-necessary information that are not relevant for TIA Portal during CAx import operations. The possible unknown artifacts can be an attributes, sub attributes, internal elements and external interfaces.

Below are the two possible reactions encountering an unknown artifact during import operations:

- The unknown artifact is "known" but meaningless to TIA Portal and can be silently ignored
- The unknown artifact is "unknown" and will be ignored but reported by a warning

Whitelist - classification of known/unknown artifacts

A special meta xml file "Automation-Cax-WhiteList.xml" is used to decide whether an unknown artifact is known or unknown to TIA Portal. You can consider artifacts which are inside AutomationProject. Hence, meta will also support configuring artifacts within AutomationProject boundary.

Below are the role class types available for configuring unknown artifacts to be ignored

Whitelist Item	Description
ItemType	Represents type of role class. The possible values for ItemType are as below: <ul style="list-style-type: none">• AutomationProjectConfigurationRoleClassLib/AutomationProject• AutomationProjectConfigurationRoleClassLib/Device• AutomationProjectConfigurationRoleClassLib/DeviceItem• AutomationProjectConfigurationRoleClassLib/DeviceUserFolder• AutomationProjectConfigurationRoleClassLib/Subnet• AutomationProjectConfigurationRoleClassLib/CommunicationInterface• AutomationProjectConfigurationRoleClassLib/CommunicationPort• AutomationProjectConfigurationRoleClassLib/Node• AutomationProjectConfigurationRoleClassLib/IoSystem• AutomationProjectConfigurationRoleClassLib/TagTable• AutomationProjectConfigurationRoleClassLib/TagUserFolder• AutomationProjectConfigurationInterfaceClassLib/Channel• AutomationProjectConfigurationInterfaceClassLib/Tag
Name	Represents name of unknown attribute/subattribute

Whitelist file structure

The following example depicts the whitelist meta file for attributes:

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Copyright @ Siemens AG 2018. All rights reserved. -->
<WhiteList>
...
...
<WhiteListItem ItemType="AutomationProjectConfigurationRoleClassLib/DeviceItem">
<AttributeList>
<Attribute Name="UnknownAttribute" /> <!--This is how an attribute as white list item is
configured-->
</AttributeList>
</WhiteListItem>
...
...
</WhiteList>
```

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Copyright © Siemens AG 2018. All rights reserved. -->
<WhiteList>
...
<WhiteListItem ItemType="*>
<AttributeList>
<Attribute Name="UnknownAttribute" />
<!--This is how an attribute as white list item is configured-->
</AttributeList>
</WhiteListItem>
...
</WhiteList>
```

The following example depicts the whitelist meta file for sub attributes:

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Copyright © Siemens AG 2018. All rights reserved. -->
<WhiteList>
...
<WhiteListItem ItemType="AutomationProjectConfigurationRoleClassLib/DeviceItem">
<SubAttributeList>
<Attribute Name ="Address">
<Attribute Name ="2">
<SubAttribute Name="UnknownSubAttribute"/> <!--This is how sub attribute as white list item
is configured-->
</Attribute>
</Attribute>
</SubAttributeList>
</WhiteListItem>
...
</WhiteList>

<?xml version="1.0" encoding="utf-8" ?>
<!-- Copyright © Siemens AG 2018. All rights reserved. -->
<WhiteList>
...
<WhiteListItem ItemType="*>
<SubAttributeList>
<Attribute Name ="Address">
<Attribute Name ="2">
<SubAttribute Name="UnknownSubAttribute"/> <!--This is how sub attribute as white list item
is configured-->
</Attribute>
</Attribute>
</SubAttributeList>
</WhiteListItem>
...
</WhiteList>
```

Note

The value "*" in ItemType shall be used to ignore all the occurrences of a particular attribute/sub-attribute under all the internal elements/external interfaces inside an AML file.

The following structure example depicts the whitelist meta file for link(s):

```
<?xml version="1.0" encoding="utf-8" ?>
<!--Copyright @ Siemens AG 2018. All rights reserved.-->
<WhiteList>
...
<WhiteListItem ItemType="AutomationProjectConfigurationRoleClassLib/DeviceItem">
<LinkList>
<Link RefBaseClassPath ="Unknown external interface RefBaseClassPath"/> --><!--This is how
an link as white list item is configured--><!--
</LinkList>
</WhiteListItem>
...
</WhiteList>
```

```
<?xml version="1.0" encoding="utf-8" ?>
<!--Copyright @ Siemens AG 2018. All rights reserved.-->
<WhiteList>
...
<WhiteListItem ItemType="*">
<LinkList>
<Link RefBaseClassPath ="Unknown external interface RefBaseClassPath"/> --><!--This is how
an link as white list item is configured--><!--
</LinkList>
</WhiteListItem>
...
</WhiteList>
```

Note

- **ItemType** represents type of role class.
 - The value "*" in ItemType shall be used to ignore all the occurrences of a particular external interface under all the internal elements of an AML file.
 - **RefBaseClassPath** represents RefBaseClassPath of an unknown external interface.
-

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Copyright © Siemens AG 2018. All rights reserved. -->
<WhiteList>
...
<WhiteListItem ItemType="*">
<LinkList>
<!--This is how ModuleAssignment is white list configured-->
<Link RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
ModuleAssignment" />
</LinkList>
</WhiteListItem>
...
</WhiteList>
```

Note**ModuleAssignment** shall be ignored from all the places in AML file

The following example depicts the whitelist meta file for object(s):

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Copyright © Siemens AG 2018. All rights reserved. -->
<WhiteList>
...
<WhiteListItem ItemType="AutomationProjectConfigurationRoleClassLib/DeviceItem">
<ObjectList>
<!--This is how an object as white list item is configured-->
<Object WhiteListKey="RefBaseClassPath"
WhiteListValue="AutomationProjectConfigurationRoleClassLib/DeviceItem"/>
</ObjectList>
</WhiteListItem>
...
</WhiteList>
```

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Copyright © Siemens AG 2018. All rights reserved. -->
<WhiteList>
...
<WhiteListItem ItemType="*">
<ObjectList>
<!--This is how an object as white list item is configured-->
<Object WhiteListKey="RefBaseClassPath"
WhiteListValue="AutomationProjectConfigurationRoleClassLib/DeviceItem"/>
</ObjectList>
</WhiteListItem>
...
</WhiteList>
```

Note

- **ItemType** represents type of role class.
 - If we put * in ItemType then this internal element will be ignored from all place of AMLI file
 - **WhiteListKey** represents RefBaseClassPath
 - **WhiteListValue** represents role class of internal element
-

See also

- [Connecting to the TIA Portal \(Page 76\)](#)
[Opening a project \(Page 106\)](#)

8.6.16 Export/Import topology

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)
- PLC is offline.

Application

In TIA Portal, you can export the devices with its topology information to an AML file. While importing to an empty TIA Portal project, the imported device items retains the topology information.

<InternalLink> element gives link details of port to port interconnection between the device items. It appears under the common parent of the connected devices, and contains unique tag names.

Attributes of a "InternalLink" element

The following table shows the related attributes of the object for CAx import and export files:

Attribute	Handling	Comment
Name	Mandatory	The tag names are formatted as "Link to Port_n" (where n varies from 1 to the number of port to port links).
RefPartnerSideA	Mandatory	Denotes the port which are linked. Formatted as UniqueIDofPort:CommunicationPortInterface
RefPartnerSideB	Mandatory	Denotes the port which are linked. Formatted as UniqueIDofPort:CommunicationPortInterface

Example: Topology view



AML structure

The following figures shows a partial element structure of the exported AML file. It contains two unique ID for the ports in PLCs.

```
...
<InternalElement ID="e1966b52-b8b3-47b4-8866-a754ebb77648" Name="Port_1">
  <Attribute Name="Label" Attribute DataType="xs:string">
```



```
...
<InternalElement ID="75f31daf-575f-48a2-ab35-8f07a376eb1b" Name="Port_1">
  <Attribute Name="Label" Attribute DataType="xs:string">
    -
```

The <InternalLink> element contains three mandatory attributes.

```
<InternalLink Name="Link to Port_1"
  RefPartnerSideA="e1966b52-b8b3-47b4-8866-a754ebb77648:CommunicationPortInterface"
  RefPartnerSideB="75f31daf-575f-48a2-ab35-8f07a376eb1b:CommunicationPortInterface" />
```

8.6.17 Import of device with Library references

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a Project (Page 106)

Application

You can import device (station)/device item (module, sub-modules) in AML file into TIA Portal using Library references.

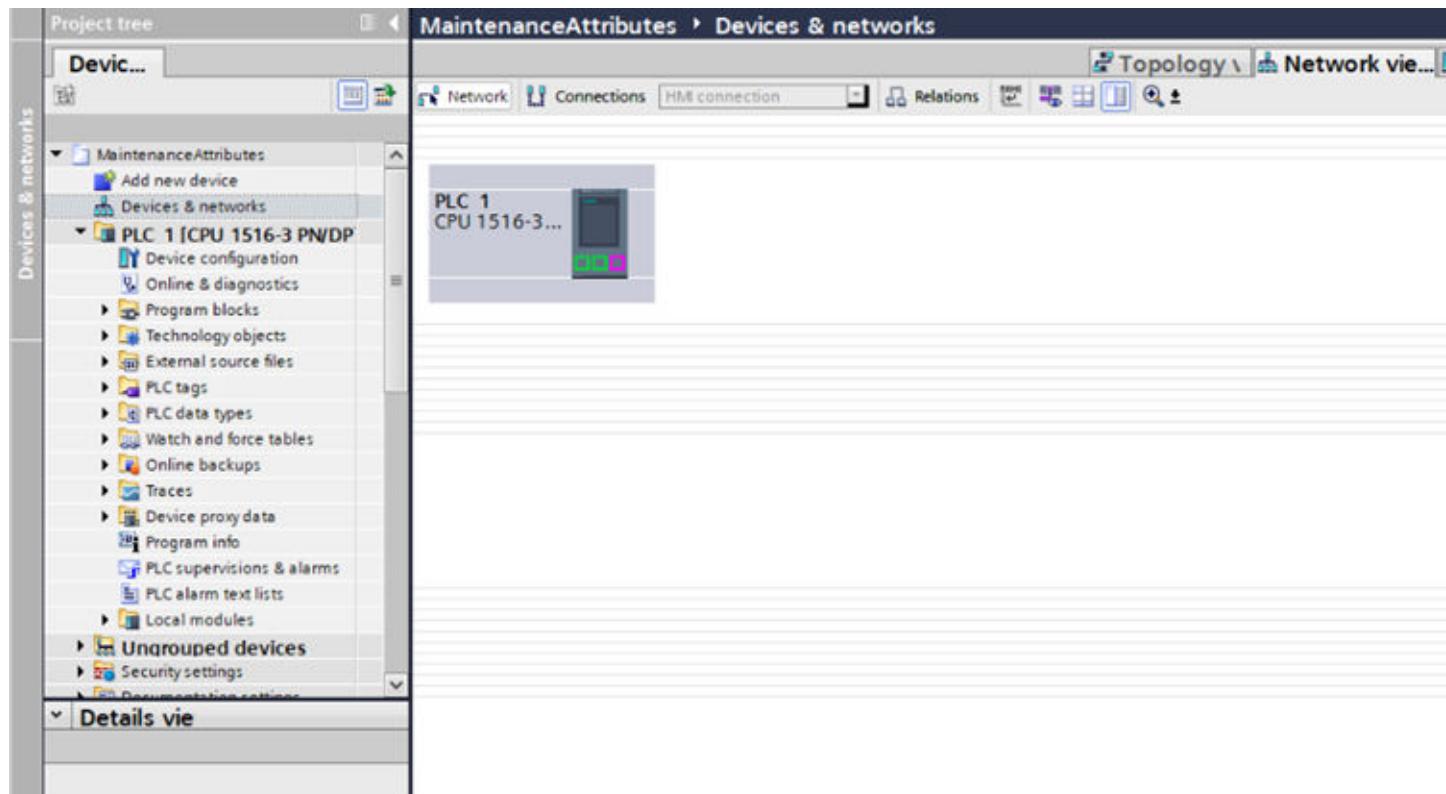
AML Structure

The following AML structure describes the XML file that shall be used during the import operations into TIA Portal project.

```
...
<InternalElement ID="bed34f88-7a3f-4e37-a32f-df1a6dcb954a" Name="S71500/ET200MP station_1">
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>System:Device.S71500</Value>
<Attribute Name="TemplateReference" AttributeDataType="xs:string">
<Value>GlobalLib://StationPlcLibrary/Master copies/DeviceFolder/S71500ET200MP station_1</Value>
</Attribute>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Device" />
...
<InternalElement ID="5082f437-4198-4dcb-b794-35b6b9fc104" Name="PLC_1">
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 214-1BE30-0XB0</Value>
<Attribute Name="TemplateReference" AttributeDataType="xs:string">
<Value>GlobalLib://StationPlcLibrary/Master copies/PLC_1</Value>
</Attribute>
</Attribute>
</InternalElement>
...
</InternalElement>
...
...
```

Example: Imported configuration

The AML snippet above corresponds to station and PLC in TIA Portal is depicted below:



See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

8.6.18 Export of a device object

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)
- The PLC is offline.

Application

The `Device` object is a container object for a central or distributed configuration. It is the parent object for `DeviceItem` objects and the top level internal element of the instance hierarchy of the TIA Portal project in between an AML file's structure.

The CAx data export supports the following types of devices specified via the AML type identifier:

- Physical modules
- GSD/GSDML based devices
- Systems

Devices can be grouped in a `DeviceUserFolder` object.

Note

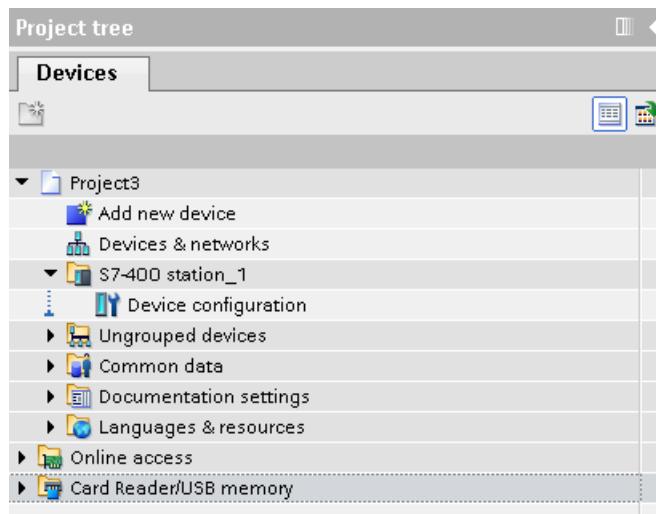
Export of a single device also exports all subnets in the project.

Attributes

The following table shows the related attributes of the device object for CAx import and export files:

Attribute	Handling	Comment
Name	Mandatory	
TypeIdentifier	Mandatory for export	Optional for import starting from AR APC V1.1
Comment	Optional	Default: ""

Example: Exported Configuration



AML structure of the export file

The following structure example depicts the export of the single device "S7-400 station_1" without racks and modules:

```

...
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
  <InternalElement ID="288b7850-688e-43b3-941e-d615ba900a02" Name="Project3">
    <Attribute Name="ProjectManufacturer" AttributeDataType="xs:string" />
    <Attribute Name="ProjectSign" AttributeDataType="xs:string" />
    <Attribute Name="ProjectRevision" AttributeDataType="xs:string" />
    <Attribute Name="ProjectInformation" AttributeDataType="xs:string" />
    <InternalElement ID="57611cfcd-6da4-444e-ac78-5fbcea20a4e1" Name="S7-400 station_1">
      <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
        <Value>System:Device.S7400</Value>
      </Attribute>
      <Attribute Name="Comment" AttributeDataType="xs:string">
        <Value>S7400 station</Value>
      </Attribute>
      <SupportedRoleClass RefRoleClassPath=
        "AutomationProjectConfigurationRoleClassLib/Device" />
    </InternalElement>
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/AutomationProject" />
  </InternalElement>
</InstanceHierarchy>
</CAEXfile>
```

Device with no type identifier and generic type identifier

The CAx import shall be able to handle devices having no type Identifier information or having generic type identifiers ('System:Device.Generic'). It shall be possible that AMI file contains certain type of device with no type Identifier information or generic type identifier ('System:Device.Generic').

But CAx import shall handle these as well to create proper devices.

The following devices support generic device or no type identifier substitution:

- GSD and GSDML devices
- Devices based on MDD (Non-GSD/GSDML devices)

For Generic device or device having no type identifier and Generic Rack, type identifier replacement , it is mandatory that the head module (in case of decentral devices) or PLC (in case of central devices) has to be present in the rack described in the AML file, otherwise the device with no type identifiers or generic device and generic rack type identifier substitution shall fail.

8.6 Importing/exporting hardware data

The following XML structure shows a device configuration with non generic type identifier:

```
<InternalElement ID="04f5d5f08-316a-4a1d-9290-9bfd75b2b2ca" Name="S7-400 station_1">
  <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
    <Value>System:Device.S7400</Value>
  </Attribute>
  <Attribute Name="Comment" AttributeDataType="xs:string">
    <Value>S7400 station</Value>
  </Attribute>
  <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
Device"/>
</InternalElement>
```

The following XML structure shows a device configuraton with generic type identifier:

```
<InternalElement ID="04f5d5f08-316a-4a1d-9290-9bfd75b2b2ca" Name="S7-400 station_1">
  <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
    <Value>System:Device.Generic</Value>
  </Attribute>
  <Attribute Name="Comment" AttributeDataType="xs:string">
    <Value>S7400 Device</Value>
  </Attribute>
  <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
Device"/>
</InternalElement>
```

The following XML structure shows a device configuration with type identifier available for device.

```
<InternalElement ID="a887601f-3ced-4f50-88ff-a9ec6eabb682" Name="S7-400 station_1">
  <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
    <Value>System:Device.S7400</Value>
  </Attribute>
  <Attribute Name="Comment" AttributeDataType="xs:string">
    <Value>S7400 Device</Value>
  </Attribute>
  <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
Device"/>
</InternalElement>
```

The following XML structure shows a device configuration with No Type Identifier available for device.

```
<InternalElement ID="a887601f-3ced-4f50-88ff-a9ec6eabb682" Name="S7-400 station_1">
  <Attribute Name="Comment" AttributeDataType="xs:string">
    <Value>S7400 Device</Value>
  </Attribute>
  <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
Device"/>
</InternalElement>
```

See also

Structure of the CAx data for importing/exporting (Page 893)

AML type identifiers (Page 898)

8.6.19 Import of a device object

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- The PLC is offline.

Application

The `Device` object is a container object for a central or distributed configuration. It is the parent object for `DeviceItem` objects and the top level internal element of the instance hierarchy of the TIA Portal project in between an AML file's structure.

The CAx data import supports the following types of devices specified via the AML type identifier:

- Physical modules
- GSD/GSDML based devices
- Systems
- Generic devices

If a `DeviceUserFolder` object exists in the TIA Portal project, the devices will be grouped in the respective folder.

If you only know the identification (`TypeIdentifier`) of a head module or a PLC and not of the respective rack and device, you can import a generic rack.

Example: `TypeIdentifier = System:<Prefix>.Generic`

For generic device replacement, the following elements have to be present in the rack described in the AML file:

- Central devices: PLC
- Decentral devices: Head module

If devices are generic, the attribute `BuiltIn` defines the kind of rack or module:

- `physical:BuiltIn = True`
- `generic:BuiltIn = False`

Example: Importing a generic device

The following structure example depicts the import of the generic "S7-400 station" device without racks and modules.

```
...
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
<InternalElement ID="d4dc896a-f4a5-41b6-9c48-8d3a0a5a4343" Name="MyProject">
  <Attribute Name="ProjectManufacturer" AttributeDataType="xs:string" />
  <Attribute Name="ProjectSign" AttributeDataType="xs:string" />
  <Attribute Name="ProjectRevision" AttributeDataType="xs:string" />
  <Attribute Name="ProjectInformation" AttributeDataType="xs:string" />
  <InternalElement ID="3e6277d1-1b12-4c18-b00e-25e3eac3ac35" Name="S7400 station_1">
    <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
      <Value>System:Device.Generic</Value>
    </Attribute>
    <Attribute Name="Comment" AttributeDataType="xs:string">
      <Value>S7400 station_1</Value>
    </Attribute>
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/Device" />
  </InternalElement>
  ...
  <SupportedRoleClass RefRoleClassPath=
    "AutomationProjectConfigurationRoleClassLib/AutomationProject" />
</InternalElement>
</InstanceHierarchy>
</CAEXFile>
```

Example: Importing a device user folder hierarchy

The following structure example depicts the import of a folder hierarchy.

```
...
<InternalElement ID="4fe37f4f-2661-4492-95f0-3d8a8160c851" Name="Project1">
    <Attribute Name="ProjectManufacturer" AttributeDataType="xs:string" />
    <Attribute Name="ProjectSign" AttributeDataType="xs:string" />
    <Attribute Name="ProjectRevision" AttributeDataType="xs:string" />
    <Attribute Name="ProjectInformation" AttributeDataType="xs:string" />
    <InternalElement ID="1ee1615f-9c67-432d-a7cc-b795babf67b6" Name="Group_1">
        <SupportedRoleClass RefRoleClassPath=
            "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
    </InternalElement>
    <InternalElement ID="ce14c85a-28de-41aa-ad08-2eb7d0fb755f" Name="Group_2">
        <InternalElement ID="852347e8-3c48-4eb9-8bd8-349d0c7caf34" Name="Group_3">
            <SupportedRoleClass RefRoleClassPath=
                "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
        </InternalElement>
        <SupportedRoleClass RefRoleClassPath=
            "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
    </InternalElement>
    <InternalElement ID="97cf7924-1756-4e32-8716-ac18990e4762" Name="Group_4">
        <SupportedRoleClass RefRoleClassPath=
            "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
    </InternalElement>
    <SupportedRoleClass RefRoleClassPath=
        "AutomationProjectConfigurationRoleClassLib/AutomationProject" />
</InternalElement>
...

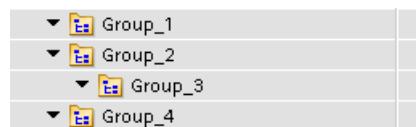
```

Imported user folder hierarchy

The name of the folders for ungrouped and unassigned devices is language specific. It is recommended to perform an import with the same user interface language as the export. Otherwise ungrouped and unassigned devices will be imported into folders named according to the export language.

For example, if you have exported a project which contains Device System Group "Ungrouped devices" in English language and then import the AML file in German language. You will see that project should have a "Nicht gruppierte Geräte" (German language) exists in device system group and have "Ungrouped device" created as user group while CAx import.

The following hierarchy is imported into the project navigation:



See also

[Structure of the CAx data for importing/exporting \(Page 893\)](#)

[AML type identifiers \(Page 898\)](#)

8.6.20 Export/Import of device with set address

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- PLC is offline.

Application

In TIA Portal, you can export the address objects of IO device items to an AML file. While importing to an empty TIA Portal project, the imported device items retains the address objects in its respective IO device items.

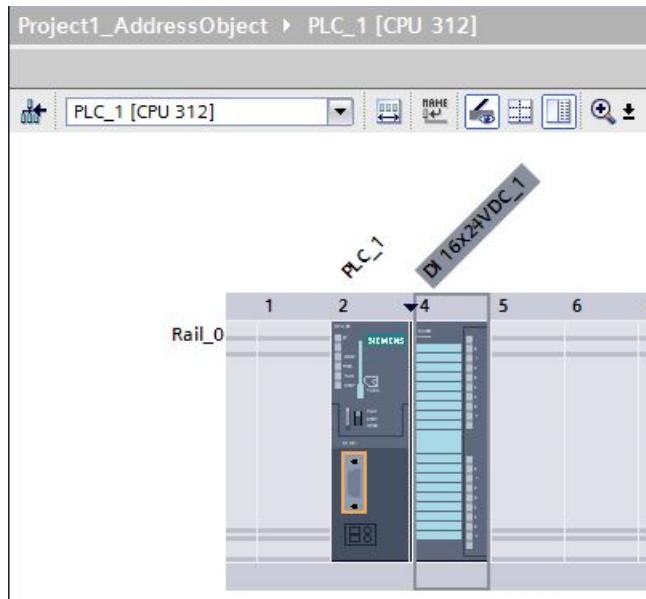
The Address attribute in the AML file contains RefSemantic mandatorily set to the specified value named `OrderedListType`.

Attributes of a "Address" element

The following table shows the related attributes of the object for CAx import and export files:

Attrib- ute	Han- dling	Comment
Io- Type	Mand- atory	Input or Output
Lengt h	Op- tional	Channel width
Star- tAd- dress	Mand- atory	Start address of the IO device.

Example: IO device items with address objects



AML Structure

The following figures shows a partial element structure of the exported AML file. It contains the Address elements and its attributes.

```

<Attribute Name="Address">
  <RefSemantic CorrespondingAttributePath="OrderedListType" />
<Attribute Name="1">
  <Attribute Name="StartAddress" AttributeDataType="xs:int">
    <Value>0</Value>
  </Attribute>
  <Attribute Name="Length" AttributeDataType="xs:int">
    <Value>16</Value>
  </Attribute>
  <Attribute Name="IoType" AttributeDataType="xs:string">
    <Value>Input</Value>
  </Attribute>
</Attribute>
</Attribute>

```

Pruned XML

Pruning is the act of optimizing the content by removing certain things which are not necessarily to be provided in the XML. In this reduced xml, the auto created sub module information are not be provided, and its corresponding address object are provided under the immediate parent.

The following figure shows a partial element structure of the exported AML file before pruning.

8.6 Importing/exporting hardware data

```
<InternalElement ID="5511a117-42c6-44b7-be5d-0f33cd46e932" Name="AS-i Master_1">
  <Attribute Name="PositionNumber" AttributeDataType="xs:int">
    <Value>4</Value>
  </Attribute>
  <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
    <Value>True</Value>
  </Attribute>
  <Attribute Name="Address">
    <RefSemantic CorrespondingAttributePath="OrderedListType" />
    <Attribute Name="1">
      <Attribute Name="StartAddress" AttributeDataType="xs:int">
        <Value>20</Value>
      </Attribute>
      <Attribute Name="Length" AttributeDataType="xs:int">
        <Value>256</Value>
      </Attribute>
      <Attribute Name="IoType" AttributeDataType="xs:string">
        <Value>Input</Value>
      </Attribute>
    </Attribute>
  </Attribute>
</InternalElement>
```

In the pruned AML file, the sub module information like `<InternalElement>` element is removed and its corresponding address objects are retained.

```
<Attribute Name="Address">
  <RefSemantic CorrespondingAttributePath="OrderedListType" />
  <Attribute Name="1">
    <Attribute Name="StartAddress" AttributeDataType="xs:int">
      <Value>20</Value>
    </Attribute>
    <Attribute Name="Length" AttributeDataType="xs:int">
      <Value>256</Value>
    </Attribute>
    <Attribute Name="IoType" AttributeDataType="xs:string">
      <Value>Input</Value>
    </Attribute>
  </Attribute>
</Attribute>
```

See also

[Pruned AML \(Page 890\)](#)

8.6.21 Export/Import of device with channels

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- PLC is offline.

Application

In TIA Portal, you can export the channel objects of IO device items to an AML file. While importing to an empty TIA Portal project, the imported device items retains the channel objects in its respective IO device items.

<ExternalInterface> element represents in node and subnet internal elements, and indicates that nodes and subnets are connected.

Attributes of a "ExternalInterface" element

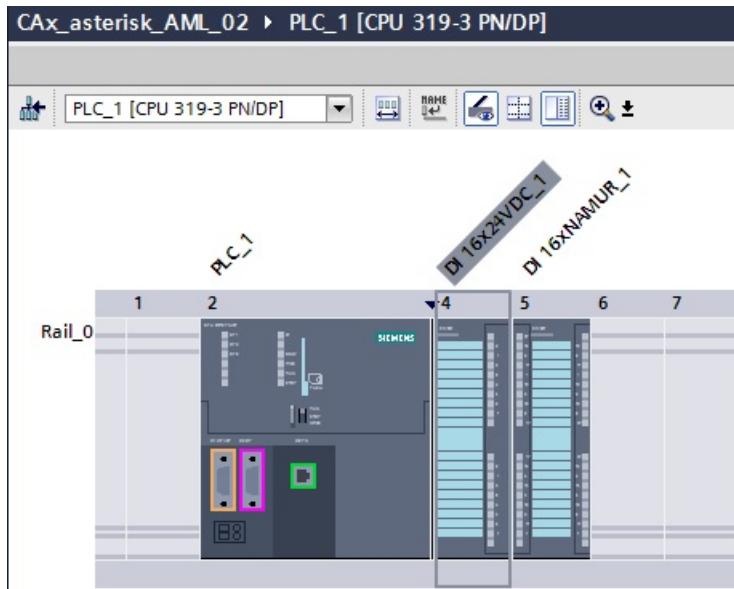
The following table shows the related attributes of the object for CAx import and export files:

Attribute	Handling	Comment
Io-Type	Mandatory	Input or Output
Length	Optional	Channel width (1 for Digital and 16 for Analog signals)
Number	Mandatory	Channel number starts from 0
Type	Mandatory	Analog or Digital

Channel numbering

Digital Input, Digital Output, Analog Input, Analog Output, and Technology channels are numbered as DI_0, DO_0, AI_0, AO_0, TO_0 respectively. Channels on the device items itself are numbered first, and channels on sub-device items are numbered subsequently (depth first). Every additional device item has its own channel numbers starting from 0.

Example: Devices with channels



AML structure

The following figures shows a partial element structure of the exported AML file.

```
<ExternalInterface ID="31ca16d3-6322-43b6-95bc-e2d7d7bfc7b7" Name="Channel_DI_0">
    RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Channel">
        <Attribute Name="Type" AttributeDataType="xs:string">
            <Value>Digital</Value>
        </Attribute>
        <Attribute Name="IoType" AttributeDataType="xs:string">
            <Value>Input</Value>
        </Attribute>
        <Attribute Name="Number" AttributeDataType="xs:int">
            <Value>0</Value>
        </Attribute>
        <Attribute Name="Length" AttributeDataType="xs:int">
            <Value>1</Value>
        </Attribute>
    </ExternalInterface>
```

8.6.22 Export of device item objects

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- The PLC is offline.

Application

The export of device item objects is only applicable for PLC devices.

`DeviceItem` objects are nested children of a `Device` object. An object of the type `DeviceItem` can be a rack or an inserted module.

- The first child item of a device is of type "rack". The `PositionNumber` of a rack starts with 0. If there are multiple racks, they are consecutively numbered (1, 2, 3, ...).
There are no restrictions about the ordering in the AML file within one hierarchy level.
- All further children of the type "rack" are modules.

The CAx data export supports the following types of device items specified via the AML type identifier:

- Physical modules
- GSD/GSDML modules

Attributes

The following table shows the related attributes of the object for CAx import and export files:

Attribute	Handling	Comment
Name	Mandatory Export-only for "BuiltIn" = TRUE	
TypeName	Export-only for "BuiltIn" = FALSE	
DeviceItemType	Export-only	Only for PLC (central devices) and device items (physical racks, modules, HeadModule). Optional during import, but any device item except Base units with DeviceItemType as accessory shall be silently ignored.
PositionNumber	Mandatory	
BuiltIn	Optional	Default: FALSE
TypeIdentifier	Mandatory for "BuiltIn" = FALSE Ignored for "BuiltIn" = TRUE	For integrated built-in device items, this attribute will be exported with its pluggable parent type identifier information and have no relevance during import and hence it is optional. For non-integrated built-in device items, this attribute is optional

Export/import

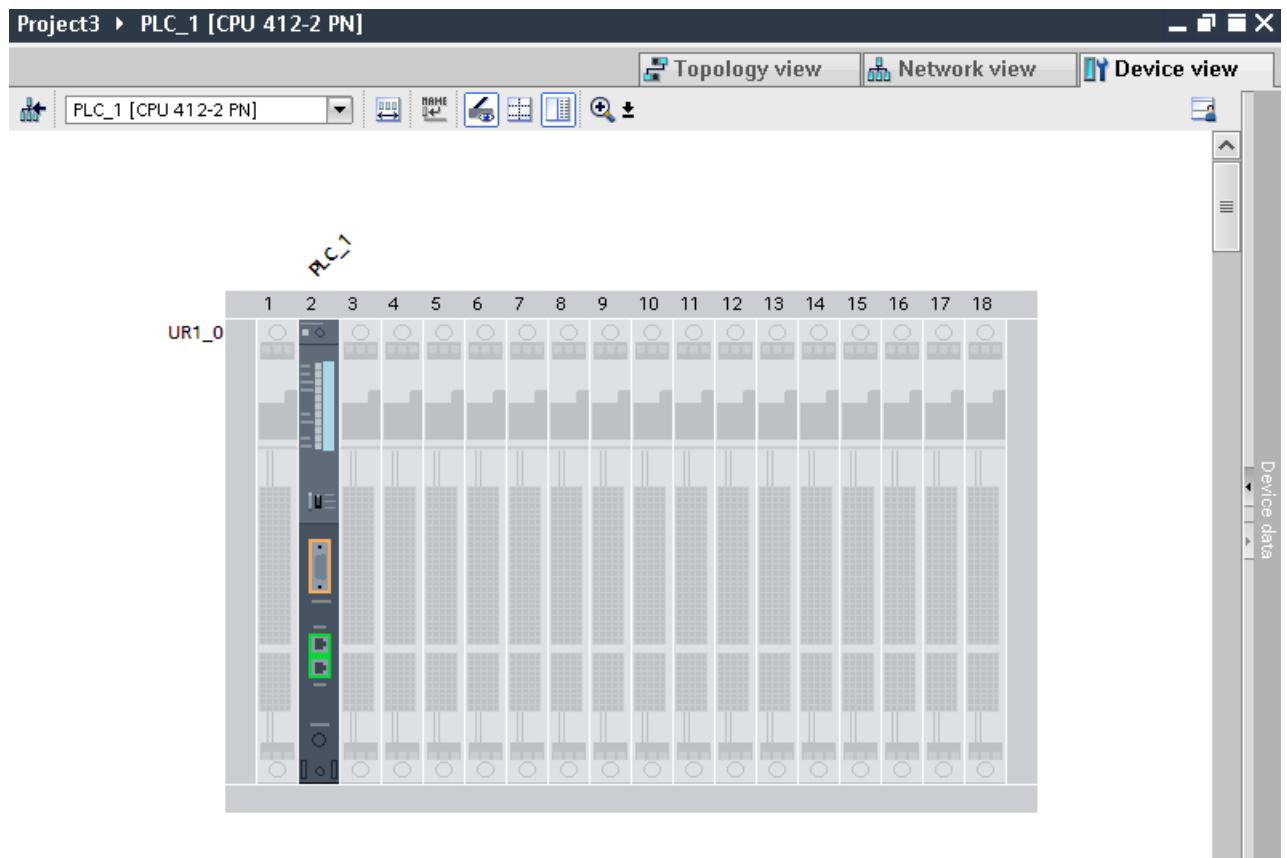
8.6 Importing/exporting hardware data

Attribute	Handling	Comment
FirmwareVersion	Optional, mandatory if the object supports firmware versions	
PlantDesignation IEC	Optional	Default: ""
LocationIdentifier IEC	Optional	Default: ""
Comment	Optional for "BuiltIn" = FALSE	Default: ""
ProductDesignation IEC	Mandatory if the object supports this attribute in TIA Portal and is non empty	Import is not supported, if the length of ProductDesignation IEC is greater than 54 characters. Export/ Import under device items as a part of AR APC V1.1.0 support for TIA Portal V16
InstallationDate	Mandatory if the object supports this attribute in TIA Portal	Export/ Import under device items as a part of AR APC V1.1.0 support for TIA Portal V16
Address	Optional	"Address" has nested attributes

The following table shows the nested attributes of the "Address" attribute of the object "DeviceItem":

Attributes for "Address"	Handling	Comment
StartAddress	Mandatory	
Length	Export-only	Export/Import of address with Length = 0 is not supported.
IoType	Mandatory	Input or Output

Example: Exported Configuration



AML structure of the export file

The following structure example depicts the export of "UR1_0" and the module "PLC_1".

```

<InternalElement ID="7624ed42-3db7-4ba5-8726-e719d7b09969" Name="S7-400_station_1">
  <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
    <Value>System:Device.S7400</Value>
  </Attribute>
  <Attribute Name="Comment" AttributeDataType="xs:string">
    <Value>S7400 station</Value>
  </Attribute>
<InternalElement ID="1ff247e6-6b94-42a2-bcd5-d673fc6e9ba5" Name="UR1_0">
  <Attribute Name="TypeName" AttributeDataType="xs:string">
    <Value>UR1</Value>
  </Attribute>
  <Attribute Name="PositionNumber" AttributeDataType="xs:int">
    <Value>0</Value>
  </Attribute>
  <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
    <Value>False</Value>
  </Attribute>
  <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
    <Value>OrderNumber:6ES7 400-1TA01-0AA0</Value>
  </Attribute>
  <Attribute Name="Comment" AttributeDataType="xs:string">
    <Value>S7 400 rack</Value>
  </Attribute>
<InternalElement ID="202421bf-a4b9-4399-a563-9f154f0eabab" Name="PLC_1">
  <Attribute Name="TypeName" AttributeDataType="xs:string">
    <Value>CPU 412-2 PN</Value>
  </Attribute>
  <Attribute Name="DeviceItemType" AttributeDataType="xs:string">
    <Value>CPU</Value>
  </Attribute>
  <Attribute Name="PositionNumber" AttributeDataType="xs:int">
    <Value>2</Value>
  </Attribute>
  <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
    <Value>False</Value>
  </Attribute>
  <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
    <Value>OrderNumber:6ES7 412-2EK06-0AB0</Value>
  </Attribute>
  <Attribute Name="Comment" AttributeDataType="xs:string">
    <Value>S7 400 plc</Value>
  </Attribute>
  <Attribute Name="FirmwareVersion" AttributeDataType="xs:string">
    <Value>V6.0</Value>
  </Attribute>
  <Attribute Name="ProductDesignation IEC" AttributeDataType="xs:string">
    <Value>Additional Information</Value>
  </Attribute>
  <Attribute Name="InstallationDate" AttributeDataType="xs:dateTime">
    <Value>2018-11-26T05:09:02.803Z</Value>
  </Attribute>
  <Attribute Name="PlantDesignation IEC" AttributeDataType="xs:string">
    <Value>PD</Value>
  </Attribute>
  <Attribute Name="LocationIdentifier IEC" AttributeDataType="xs:string">
    <Value>LI</Value>
  </Attribute>
  <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/DeviceItem" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/DeviceItem" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Device" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/AutomationProject" />
</InternalElement>

```

See also

[Export/Import of GSD/GSDML based devices and device items \(Page 963\)](#)

[Structure of the CAx data for importing/exporting \(Page 893\)](#)

[AML type identifiers \(Page 898\)](#)

8.6.23 Import of device item objects

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)
- The PLC is offline.

Application

The import of device item objects is only applicable for PLC devices.

DeviceItem objects are nested children of a Device object. An object of the type DeviceItem can be a rack or an inserted module.

- The first child item of a device has to be of type rack. The PositionNumber of a rack starts with 0. If there are multiple racks, they are consecutively numbered (1, 2, 3, ...). There are no restrictions about the ordering in the AML file within one hierarchy level.
- All further children of the type rack are modules.

The CAx data import supports the following types of device items specified via the AML type identifier:

- Physical modules
- GSD/GSDML modules
- Generic modules

If you only know the identification (TypeIdentifier) of a head module or a PLC and not of the respective rack and device, you can import a generic rack.

Example: TypeIdentifier = System:Rack.Generic

For generic rack replacement, the following elements have to be present in the rack described in the AML file:

- Central devices: PLC
- Decentral devices: Head module

A generic rack type derives from the Device type. Therefore, an AML file that is imported to TIA Portal can use this rack's type identifier:

In this case the TIA Portal determines the type identifier for the rack.

If racks and modules are generic, the attribute `BuiltIn` defines the kind of rack or module:

- `physical`: `BuiltIn = True`
- `generic`: `BuiltIn = False`

Restrictions

While importing, the attribute `DeviceItemType` has no relevance and hence it is optional.

Note

Attribute "FirmwareVersion"

If no `FirmwareVersion` is specified in the import file CAx import uses the latest firmware version which is available in the TIA Portal

If the `FirmwareVersion` attribute exists in the import file with an empty value, the device item import fails and an error message will be logged.

Example: Importing a generic device

The following structure example depicts the import of the generic rack "Rack_1".

```

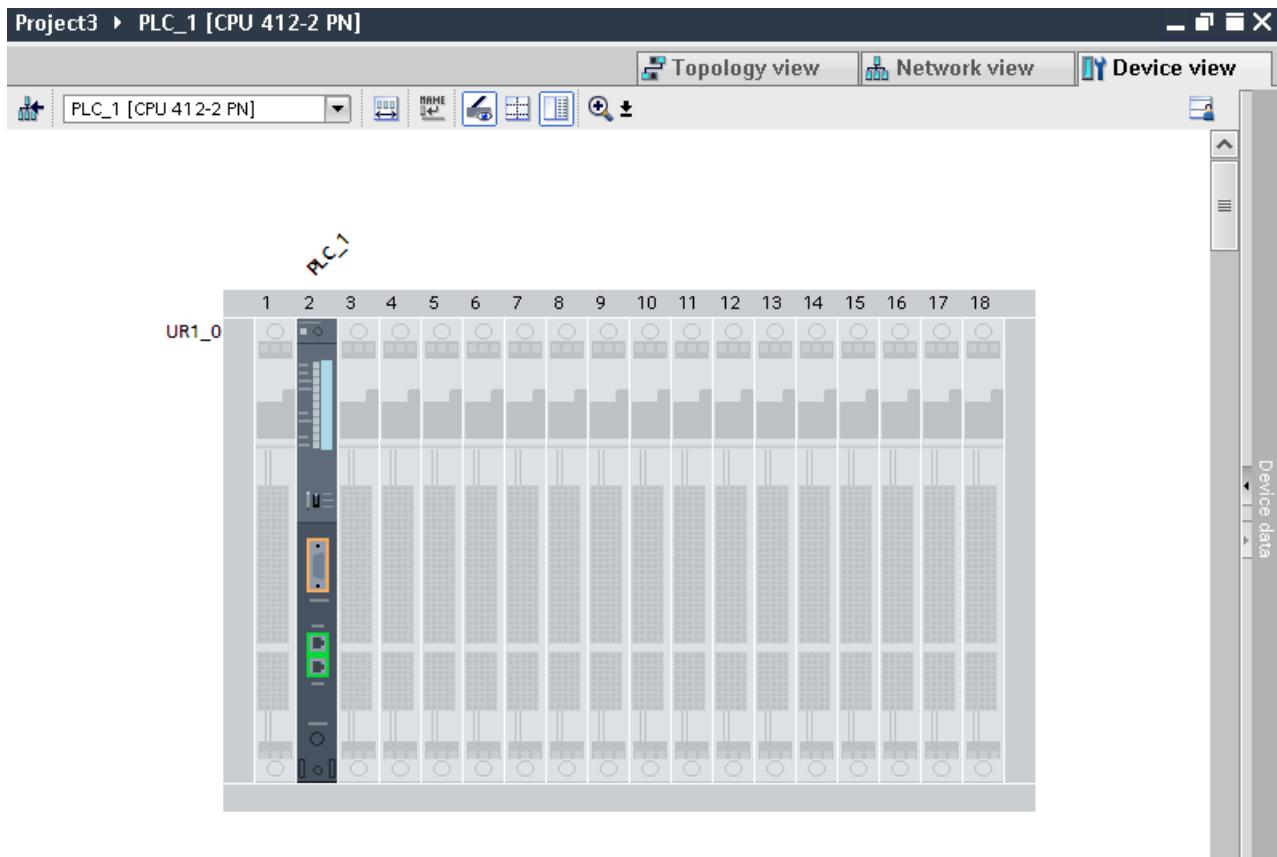
...
<InternalElement ID="6563466e-2de9-42ca-951d-eb8f2545958d" Name="S7-400_station_1">
  <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
    <Value>System:Device.S7400</Value>
  </Attribute>
  <InternalElement ID="96930368-14ec-43e2-b9b7-c1fefc4b0534" Name="UR1_0">
    <Attribute Name="TypeName" AttributeDataType="xs:string">
      <Value>UR1</Value>
    </Attribute>
    <Attribute Name="PositionNumber" AttributeDataType="xs:int">
      <Value>0</Value>
    </Attribute>
    <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
      <Value>False</Value>
    </Attribute>
    <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
      <Value>System:Rack.Generic</Value>
    </Attribute>
    <InternalElement ID="a1de449e-4f89-45af-8bbc-f77c28bccd04" Name="PLC_1">
      <Attribute Name="TypeName" AttributeDataType="xs:string">
        <Value>CPU 412-2 PN</Value>
      </Attribute>
      <Attribute Name="DeviceItemType" AttributeDataType="xs:string">
        <Value>CPU</Value>
      </Attribute>
      <Attribute Name="PositionNumber" AttributeDataType="xs:int">
        <Value>2</Value>
      </Attribute>
      <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
        <Value>False</Value>
      </Attribute>
      <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
        <Value>OrderNumber:6ES7 412-2EK06-0AB0</Value>
      </Attribute>
      <Attribute Name="FirmwareVersion" AttributeDataType="xs:string">
        <Value>V6.0</Value>
      </Attribute>

      <SupportedRoleClass RefRoleClassPath=
        "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
    </InternalElement>
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath=
    "AutomationProjectConfigurationRoleClassLib/Device" />
</InternalElement>
...

```

Imported Configuration

The following figure shows the imported configuration in the TIA Portal user interface:



See also

[Structure of the CAx data for importing/exporting \(Page 893\)](#)

[AML type identifiers \(Page 898\)](#)

8.6.24 Export/Import of GSD/GSDML based devices and device items

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
[See Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
[See Opening a project \(Page 106\)](#)
- PLC is offline.

Application

The CAx Import/Export of GSD/GSDML based devices and device items is similar to the import/export of standard devices.

For GSD/GSDML based devices and device items the exportable attributes differ, e. g. for GSD/GSDML the attribute `Label` exists.

Generic import of devices and racks are possible. For the import, you use the same identifier as for standard devices:

- Import of a generic device: `TypeIdentifier = System:Device.Generic`
- Import of a generic rack: `TypeIdentifier = System:Rack.Generic`

If devices are generic, the attribute `BuiltIn` defines the kind:

- `physical: BuiltIn = True`
- `generic: BuiltIn = False`

Attributes for a device

The following table shows the related attributes of device for CAx import and export files:

Attribute	Handling for attribute	Comment
Name	Mandatory for export and import	
TypeIdentifier	Mandatory for export and optional for import starting from AR APC V1.1	
Comment	Optional for import	

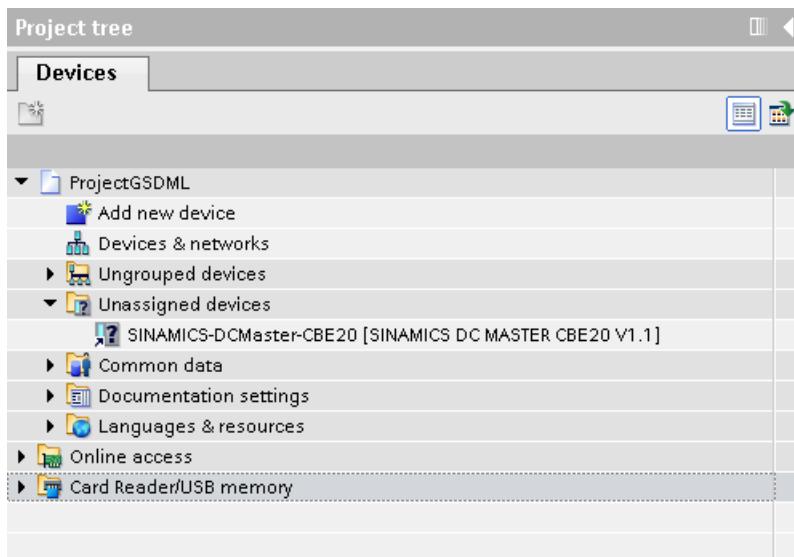
Attributes for a device item

The following table shows the related attributes of a device item for CAx import and export files:

Attribute	Handling for attribute <code>BuiltIn = FALSE</code> Generic device items	Handling for attribute <code>BuiltIn = TRUE</code> Physical device items	Comment
Name	Mandatory	Export-only	
TypeName	Export-only	Not applicable	
DeviceItem-Type	Export-only	Export-only	Only for PLC (central devices) und Head-Module (decentral devices) device items Optional during import, but any device item except Base units with DeviceItem-Type as accessory shall be ignored.
PositionNumber	Mandatory	Mandatory for export Exceptional cases: Device item type interface: Optional for import Device item type port: Optional	
BuiltIn	Optional		Default: FALSE

Attribute	Handling for attribute BuiltIn = FALSE Generic device items	Handling for attribute BuiltIn = TRUE Physical device items	Comment
TypeIdentifier	Mandatory for "BuiltIn" = FALSE	Ignored for "BuiltIn" = TRUE	For integrated built-in device items, this attribute will be exported with its pluggable parent type identifier information. This attribute has no relevance during import, hence it is optional. For non-integrated built-in device items, this attribute has no relevance.
Comment	Optional	-	
Label	-	- Device item type interface: Mandatory Device item type port: Mandatory	

Example: Exported GSD/GSDML device



AML structure of the export file

The following figure shows the structure of the exported AML file.

```
...
<InternalElement ID="9ae02cde-dfb4-4d43-a649-68b9ede7fc3d" Name="Ungrouped devices">
  <InternalElement ID="12d4ce0f-346d-4bfa-b139-c9d0db64c794" Name="GSD device_1">
    <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
      <Value>GSD:GSDML-V2.31-SIEMENS-SINAMICS_DCMaster-20140704.XML/D</Value>
    </Attribute>
  <InternalElement ID="ccb1cb62-67b2-4b8c-951f-10c7ffb4d787" Name="Rack">
    <Attribute Name="TypeName" AttributeDataType="xs:string">
      <Value>Rack</Value>
    </Attribute>
  ...
  <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
    <Value>GSD:GSDML-V2.31-SIEMENS-SINAMICS_DCMaster-20140704.XML/R/IDD_14</Value>
  </Attribute>
  <InternalElement ID="74f25b5c-0c09-46d0-9011-f341a3e98a0d" Name="SINAMICS-DCMaster-CBE20">
    <Attribute Name="TypeName" AttributeDataType="xs:string">
      <Value>SINAMICS DC MASTER CBE20 V1.1</Value>
    </Attribute>
    <Attribute Name="DeviceItemType" AttributeDataType="xs:string">
      <Value>HeadModule</Value>
    </Attribute>
    <Attribute Name="PositionNumber" AttributeDataType="xs:int">
      <Value>0</Value>
    </Attribute>
    <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
      <Value>False</Value>
    </Attribute>
    <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
      <Value>GSD:GSDML-V2.31-SIEMENS-SINAMICS_DCMaster-20140704.XML/DAP/IDD_14</Value>
    </Attribute>
    <InternalElement ID="94f34bb9-fe47-4904-b8a1-62e8fb6b1b74" Name="SINAMICS DC MASTER CBE20 V1.1">
      <Attribute Name="PositionNumber" AttributeDataType="xs:int">
        <Value>0</Value>
      </Attribute>
      <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
        <Value>True</Value>
      </Attribute>
      <SupportedRoleClass RefRoleClassPath=
        "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
    </InternalElement>
  ...
  <SupportedRoleClass RefRoleClassPath=
    "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath=
  "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath=
  "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
</InternalElement>
...

```

Generic & Non generic rack with & without Typelidentifier

The CAx import shall be able to handle devices with no type identifier information or generic type identifiers i.e.'System:Device.Generic' and racks with generic type identifiers i.e. 'System:Rack.Generic'.

While importing it shall be possible that AML file contains certain types of devices with no type identifier or generic type identifier i.e 'System:Device.Generic' and rack device items with generic type identifier i.e, 'System:Rack.Generic'. But CAx import shall handle these as well to create proper devices and rack device items.

Following devices support generic device , device with no type identifier and generic rack substitution:

- GSD and GSDML devices - All the GSD/GSDML devices with GSD/GSDML racks.
- Devices based on MDD (Non-GSD/GSDML devices) - The devices with system rack type identifiers.

CAx export has no relevance with generic rack handling. CAx always exports non generic rack type identifiers.

For Generic device or device having no type identifier and Generic Rack, type identifier replacement , it is mandatory that the head module (in case of decentral devices) or PLC (in case of central devices) has to be present in the rack described in the AML file, otherwise the device with no type identifiers or generic device and generic rack type identifier substitution shall fail.

The following XML structure shows a GSDML device configuration with device (with Type Identifier) and rack (with Typelidentifier):

```

<InternalElement ID="bc3b50fa-5cfa-4bf3-a496-8e46080d4f86" Name="GSD_device_1">
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>GSD:GSDML-V2.32-SIEMENS-SINAMICS_DCMASTER-20160531.XML/D</Value>
</Attribute>
<InternalElement ID="c80f2d97-66c9-4f31-bf6b-17e3e0de0509" Name="Rack">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>Rack</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>0</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>GSD:GSDML-V2.32-SIEMENS-SINAMICS_DCMASTER-20160531.XML/R/IDD_14</Value>
</Attribute>
<InternalElement ID="f519b4b9-b1f7-4011-bed2-25be85c8c2a8" Name="SINAMICS-DCMaster-CBE20">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>SINAMICS DC MASTER CBE20 V1.1</Value>
</Attribute>
<Attribute Name="DeviceItemType" AttributeDataType="xs:string">
<Value>HeadModule</Value>
</Attribute>
```

8.6 Importing/exporting hardware data

The following XML structure shows a GSDML device configuration with device (no type identifier) and rack (generic type identifier):

```
<InternalElement ID="bc3b50fa-5cfa-4bf3-a496-8e46080d4f86" Name="GSD device_1">
<InternalElement ID="c80f2d97-66c9-4f31-bf6b-17e3e0de0509" Name="Rack">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>Rack</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>0</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>System:Rack.Generic</Value>
</Attribute>
<InternalElement ID="f519b4b9-b1f7-4011-bed2-25be85c8c2a8" Name="SINAMICS-DCMaster-CBE20">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>SINAMICS DC MASTER CBE20 V1.1</Value>
</Attribute>
<Attribute Name="DeviceItemType" AttributeDataType="xs:string">
<Value>HeadModule</Value>
</Attribute>
```

The following XML structure shows a GSDML device configuration with device (generic type identifier) and rack (generic type identifier):

```

<InternalElement ID="bc3b50fa-5cfa-4bf3-a496-8e46080d4f86" Name="GSD device_1">
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>System:Device.Generic</Value>
</Attribute>
<InternalElement ID="c80f2d97-66c9-4f31-bf6b-17e3e0de0509" Name="Rack">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>Rack</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>0</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>System:Rack.Generic</Value>
</Attribute>
<InternalElement ID="f519b4b9-b1f7-4011-bed2-25be85c8c2a8" Name="SINAMICS-DCMaster-CBE20">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>SINAMICS DC MASTER CBE20 V1.1</Value>
</Attribute>
<Attribute Name="DeviceItemType" AttributeDataType="xs:string">
<Value>HeadModule</Value>
</Attribute>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>GSD:GSDML-V2.32-SIEMENS-SINAMICS_DCMMASTER-20160531.XML/DAP/IDD_14</Value>
</Attribute>
```

See also

[AML type identifiers \(Page 898\)](#)

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

8.6.25 Export/Import device configuration with virtual interface

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)
- PLC is offline.

Application

In TIA Portal, ports are used for inter-device communication and found under interfaces. But there are certain kind of devices where the ports are found directly under the device items which are not interfaces. This is not in accordance with the AML standards where ports are always looked for under interfaces.

CAx shall use an imaginary interface referred to Virtual interface to export and import device configuration in case ports are found directly under non-interface device items.

Export of AML file

The below example shows an AML file having virtual interface:

```

<InternalElement ID="822622a0-056a-494c-a802-2463c5e1b47d" Name="SCALANCE interface_1">
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<InternalElement ID="5a604a57-bc2d-4763-8df0-7d20000faf1b" Name="VirtualInterface_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X1</Value>
</Attribute>
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>Ethernet</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<InternalElement ID="3c4862a1-b8ed-4610-88f3-29dc8328e748" Name="Port_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>P1</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<InternalElement ID="220dd49d-8d23-44b1-bdc1-878516540313" Name="Port_2">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>P2</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>2</Value>
</Attribute> <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="99c6253b-c546-4720-af54-92db926b8231"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
```

8.6 Importing/exporting hardware data

The virtual interface shall be exported with the following attributes till before TIA Portal v16 :

- Name as ScalanceInterface_1
- Label as Switch

From TIA Portal v16 the virtual interface shall be exported with the below attributes

- Name as VirtualInterface_1
- Label as X1
- Type as Ethernet

Import

CAx shall support import of virtual interfaces. In this case, it is expected to process the ports under the Virtual interface in the AML file under the right parent in the TIA Portal. Here any interface with Label as switch is to be considered as virtual interface. But with TIA Portal v16, Label is no more identifier as it is changed to X1, which is just like a real interface.

Type attribute although is set to "Ethernet" for virtual interface and it is optional. So with TIA Portal v16 all interface elements in AML file not labelled to "Switch" shall be treated in a general way, and whenever encountered in AML file CAx shall try to fetch from TIA Portal.

In case of Virtual interface, CAx shall fail to find it but process the ports inside it. But in case of real interface if CAx doesn't find it, the ports inside it shall not be processed

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

8.6.26 Export/Import of subnets

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)
- PLC is offline.

AML structure

Subnets describe a physical network especially which devices are connected to the same network of type PROFIBUS, PROFINET, MPI or ASI.

The link between a network and the device items are modeled as a reference to the network object. There is no reference from the network object to the attached device items. The network parameters are stored in the network object. The parameters concerning a network interface of

a given device item, attached to a network, are stored in a net node object in that device item. The communication is often regulated using “channels”, “ports” and “interfaces”.

Subnets are exported as internal elements of the role class "Subnet" in the instance hierarchy in the AML file.

A subnet has the following related elements in the AML structure:

- Internal element of the role class "Node"
Defines the interface on a device item.
- <InternalLink>
Defines the connected partners of the subnet. <InternalLink> tags name is unique and is always added under the project's internal element in the AML file.

```
....  
<SupportedRoleClass RefRoleClassPath=  
    "AutomationProjectConfigurationRoleClassLib/AutomationProject" />  
<InternalLink Name="Link To Port_1"  
    RefPartnerSideA="1e3e4c5b-04c1-4d2c-9aee-cad53cc92dba:CommunicationPortInterface"  
    RefPartnerSideB="d45aa36a-a7f2-4862-a266-d6727b9cf75:CommunicationPortInterface" />  
<InternalLink Name="Link To Subnet_1"  
    RefPartnerSideA="beb4eb8e-1a45-45ce-a703-1acfac73e5f3:LogicalEndPoint_Node"  
    RefPartnerSideB="1062a384-d3ca-4183-9ac2-0934a5ab7286:LogicalEndPoint_Subnet" />  
<InternalLink Name="Link To Subnet_2"  
    RefPartnerSideA="a3e85aed-580a-45c8-943e-da7de8280b7c:LogicalEndPoint_Node"  
    RefPartnerSideB="1062a384-d3ca-4183-9ac2-0934a5ab7286:LogicalEndPoint_Subnet" />  
</InternalElement>  
</InstanceHierarchy>  
</CAEXFile>
```

- <ExternalInterface>
Represents in node and subnet internal elements that nodes and subnets are connected.
If the nodes or subnets are not connected then the <ExternalInterface> element for node and subnet do not exist.

```
...  
<InternalElement ID="1062a384-d3ca-4183-9ac2-0934a5ab7286" Name="PN/IE_1">  
  <Attribute Name="Type" AttributeDataType="xs:string">  
    <Value>Ethernet</Value>  
  </Attribute>  
  <ExternalInterface ID="55ecf2cb-e72a-4974-8ed0-1d4e1ade3509"  
    Name="LogicalEndPoint_Subnet"  
    RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />  
  <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Subnet" />  
</InternalElement>  
...
```

Application

In TIA Portal V16, CAx import supports Ethernet and Profinet and export will always be Ethernet for Subnet Type and Node attributes.

The CAx Import/Export supports the following types of subnets:

- Ethernet
- PROFIBUS

8.6 Importing/exporting hardware data

- MPI
- ASi

Note

- Cax import will support Ethernet and Profinet for Subnet Type and Node Type attribute.
 - CAx import of an AML file with 'case invariant' string values for the attributes (For ex: Profinet, PROFINET, pROFINET, ProFINET etc.,) should be supported.
 - Cax export will always be Ethernet.
-

Attributes of a "Subnet" element

The following table shows the related attributes of the object for CAx import and export files:

Attribute	Handling	Comment
Name	Mandatory	
Type	Mandatory	Ethernet or PROFIBUS or MPI or ASI

Attributes of "CommunicationInterface" elements

The following table shows the related attributes of the objects for CAx import and export files:

Attribute	Handling	Comment
Name	Mandatory	No relevance for "fixed" device items.
Label	Mandatory	Label may be missing if "BuiltIn" = TRUE and "PositionNumber" are specified for the related "DeviceItem" object.
TypeIdentifier	Mandatory	For integrated built-in device items, this attribute shall be exported with its pluggable parent type identifier information. And during import this attribute has no relevance. For non-integrated built-in device items, this attribute has no relevance.
FirmwareVersion	Mandatory	
TypeName	Export-only	No relevance for "BuiltIn" device items.
DeviceItemType	Export-only	Only for CPU and HeadModule This attribute is optional during import but any device item except Base units with DeviceItemType as Accessory shall be silently ignored.
PositionNumber	Mandatory	No relevance for the import of "BuiltIn" device items.
BuiltIn	Mandatory for export Optional for import	No relevance for the import of "Non-BuiltIn" device items. False by default for import.
Comment	Optional	Not applicable for "BuiltIn" device items.

Attributes of "CommunicationPort" elements

The following table shows the related attributes of the objects for CAx import and export files:

Attribute	Handling	Comment
Name	Mandatory	No relevance for "BuiltIn" device items.
Label	Mandatory	<p>Label may be missing if "BuiltIn" = TRUE and "PositionNumber" are specified for the related "DeviceItem" object.</p> <p>While importing Port label value, comparison of label value shall be case invariant (For ex: p1 r, P1 R, P1, p1 etc.,).</p> <p>For Ports enabled for ring topology, indicated with suffix "R", import shall be treated "equivalent" to Ports having label values without suffix.</p> <p>For example: The following port labels are treated as equal:</p> <ul style="list-style-type: none"> • P1 equals P1 • P1 equals P1 R • P1 R equals P1 • P1 R equals P1 R
TypeIdentifier	Mandatory	<p>For integrated built-in device items, this attribute shall be exported with its pluggable parent type identifier information. And during import this attribute has no relevance.</p> <p>For non-integrated built-in device items, this attribute has no relevance</p>
FirmwareVersion	Mandatory	
TypeName	Export-only	No relevance for "BuiltIn" device items.
DeviceItemType	Export-only	<p>Only for CPU and HeadModule.</p> <p>Optional during import, but any device item except Base units with DeviceItemType as accessory shall be silently ignored.</p>
PositionNumber	Mandatory	<p>Only relevant for the import of "BuiltIn" device items, if "Label" attribute is not specified.</p> <p>If both "PositionNumber" and "Label" are configured, then "PositionNumber" gets higher precedence.</p>
BuiltIn	Mandatory for export Optional for import	<p>No relevance for the import of "Non-BuiltIn" device items.</p> <p>False by default for import.</p>
Comment	Optional	Not applicable for "BuiltIn" device items.

Attributes of a "Node" element

The following table shows the related attributes of the object for CAx import and export files:

Attribute	Handling	Comment
Name	Export-only	MPI, PROFIBUS, PROFINET
Type	Export-only	Ethernet or PROFIBUS or MPI or ASI
NetworkAddress	Mandatory	
SubnetMask	Optional	<p>PROFINET</p> <p>For import, default value is retained if no value is set.</p>
RouterAddress	Optional	<p>PROFINET</p> <p>For import, default value is retained if no value is set.</p>

8.6 Importing/exporting hardware data

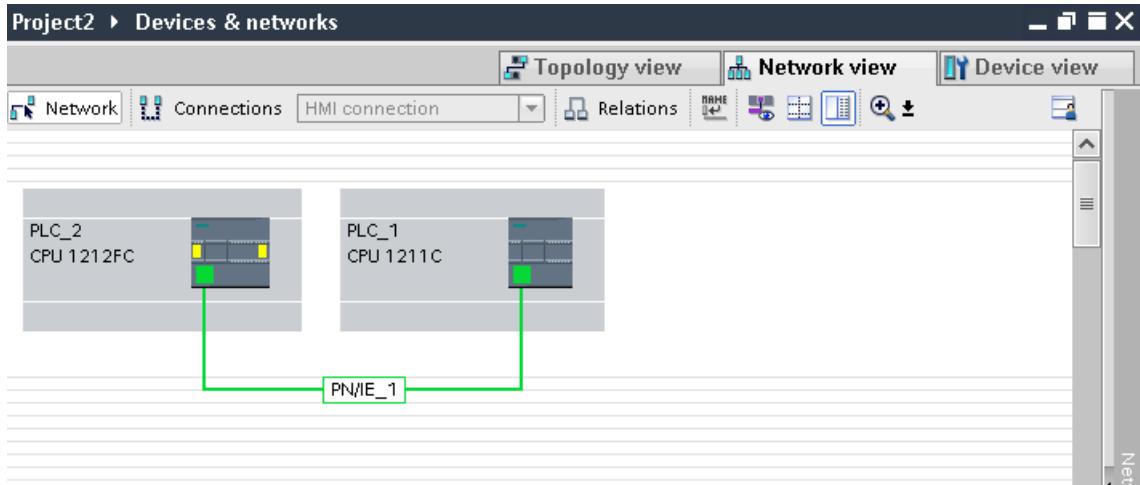
Attribute	Handling	Comment
DhcpClientId	Optional	PROFINET For import, default value is retained if no value is set.
IpProtocolSelection	Optional	PROFINET For import, default value is retained if no value is set. Values: Project, Dhcp, UserProgram, OtherPath

Attributes of a "Channel" element"

The following table shows the related attributes of the object for CAx import and export files:

Attribute	Handling	Comment
Type	Mandatory	Digital or Analog
IoType	Mandatory	Input or Output
Number	Mandatory	
Length	Export-only	

Example: Exported subnet



AML structure

The following figures show the structure of the exported AML file:

```
...
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
  <InternalElement ID="e9d2bedb-f8c1-4148-acda-c3c68836c7dd" Name="Project2">
    ...
      <InternalElement ID="1062a384-d3ca-4183-9ac2-0934a5ab7286" Name="PN/IE_1">
        <Attribute Name="Type" AttributeDataType="xs:string">
          <Value>Ethernet</Value>
        </Attribute>
        <ExternalInterface ID="55ecf2cb-e72a-4974-8ed0-1d4e1ade3509">
          Name="LogicalEndPoint_Subnet"
          RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
        <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Subnet" />
      </InternalElement>
      <InternalElement ID="b011dbb1-efa4-46c0-a26f-f9bd047cda4f" Name="S7-1200 station_1">
        ...
        <InternalElement ID="d006e41b-05ff-44ab-baab-fca15f99e86c" Name="PROFINET interface_1">
          ...
            <InternalElement ID="beb4eb8e-1a45-45ce-a703-1acfac73e5f3" Name="E1">
              ...
                <ExternalInterface ID="a365b498-20cc-4e0b-99ca-5c5257632b96">
                  Name="LogicalEndPoint_Node" RefBaseClassPath=
                  "CommunicationInterfaceClassLib/LogicalEndPoint" />
                <SupportedRoleClass RefRoleClassPath=
                  "AutomationProjectConfigurationRoleClassLib/Node" />
              </InternalElement>
              <InternalElement ID="d45aa36a-a7f2-4862-a266-d6727b9cf75" Name="Port_1">
                ...
                  <ExternalInterface ID="32c6ba4a-b01f-4678-b721-ea284779e96c">
                    Name="CommunicationPortInterface"
                    RefBaseClassPath=
                    "AutomationProjectConfigurationInterfaceClassLib/CommunicationPortInterface" />
                  <SupportedRoleClass RefRoleClassPath=
                    "AutomationProjectConfigurationRoleClassLib/CommunicationPort" />
                </InternalElement>
                <SupportedRoleClass RefRoleClassPath=
                  "AutomationProjectConfigurationRoleClassLib/CommunicationInterface" />
              </InternalElement>
              <SupportedRoleClass RefRoleClassPath=
                "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
            </InternalElement>
            <SupportedRoleClass RefRoleClassPath=
              "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
          </InternalElement>
          <SupportedRoleClass RefRoleClassPath=
            "AutomationProjectConfigurationRoleClassLib/Device" />
        </InternalElement>
      ...

```

```

...
<InternalElement ID="7cf0ea2b-b66f-4ad4-8a03-5a8691cbe04d" Name="PLC_2">
...
<InternalElement ID="b287020d-667b-483d-a8e0-c5466ac2f5c3" Name="PROFINET interface_1">
  <Attribute Name="Label" AttributeDataType="xs:string">
    <Value>X1</Value>
  </Attribute>
  <InternalElement ID="a3e85aed-580a-45c8-943e-da7de8280b7c" Name="E1">
    <Attribute Name="Type" AttributeDataType="xs:string">
      <Value>Ethernet</Value>
    </Attribute>
    <Attribute Name="NetworkAddress" AttributeDataType="xs:string">
      <Value>192.168.0.2</Value>
    </Attribute>
    <Attribute Name="SubnetMask" AttributeDataType="xs:string">
      <Value>255.255.255.0</Value>
    </Attribute>
    <Attribute Name="DeviceNumber" AttributeDataType="xs:string">
      <Value>0</Value>
    </Attribute>
    <Attribute Name="IpProtocolSelection" AttributeDataType="xs:string">
      <Value>Project</Value>
    </Attribute>
    <ExternalInterface ID="6ae8eb93-09d3-4f8c-b529-a12148c71bf4"
      Name="LogicalEndPoint_Node"
      RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
    <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Node" />
  </InternalElement>
<InternalElement ID="1e3e4c5b-04c1-4d2c-9aee-cad53cc92dba" Name="Port_1">
...
  <ExternalInterface ID="1f5b2a3d-fcd1-460a-b846-30dad8726d1"
    Name="CommunicationPortInterface"
    RefBaseClassPath=
      "AutomationProjectConfigurationInterfaceClassLib/CommunicationPortInterface" />
  <SupportedRoleClass RefRoleClassPath=
    "AutomationProjectConfigurationRoleClassLib/CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath=
  "AutomationProjectConfigurationRoleClassLib/CommunicationInterface" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath=
  "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
</InternalElement>
...

```

See also

[Structure of the CAx data for importing/exporting \(Page 893\)](#)

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

8.6.27 Export/Import of IO-systems

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open.
See Opening a project (Page 106)
- PLC is offline.

AML structure

IO-system are represented in the AML structure as <InternalElement>.

IO-system of as master or IO controller are added under the element <CommunicationInterface> of an interface device item.

```
...
<InternalElement ID="[Communication Interface UniqueID]"
    Name="[Communication Interface Name]">
...
<!--Node-->
<InternalElement ID="[Node UniqueID]" Name="[Node Name]">
...
<ExternalInterface ID="[External Interface UniqueID]"
    Name="LogicalEndPoint_Node"
    RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
<SupportedRoleClass
    RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Node" />
</InternalElement>
<!--IoSystem-->
<InternalElement ID="[IoSystem UniqueID]" Name="[IoSystem Name]">
<Attribute Name="Number" AttributeDataType="xs:integer">
    <Value>[IoSystem Number]</Value>
</Attribute>
<ExternalInterface ID="[External Interface UniqueID]"
    Name="LogicalEndPoint_Interface"
    RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
<SupportedRoleClass
    RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/IoSystem" />
</InternalElement>
<SupportedRoleClass
    RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/CommunicationInterface" />
</InternalElement>
```

Connected IO-system as slave or IO-device are added as <ExternalInterface> elements under the element <CommunicationInterface> of an interface device item.

8.6 Importing/exporting hardware data

```

<InternalElement ID="[Communication Interface UniqueID]"
  Name="[Communication Interface Name]">
...
<ExternalInterface ID="[External Interface UniqueID]"
  Name="LogicalEndPoint_Interface"
  RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
<!--Node-->
<InternalElement ID="[Node UniqueID]" Name="[Node Name]">
  <ExternalInterface ID="[External Interface UniqueID]"
    Name="LogicalEndPoint_Node"
    RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
  <SupportedRoleClass
    RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Node" />
</InternalElement>
<SupportedRoleClass
  RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/CommunicationInterface" />
</InternalElement>

```

The connected partners of the IO-systems are represented as `<InternalLink>` elements. `<InternalLink>` tags are added under common parent of an IO-system and connected slave device item, e. g. Project, DeviceFolder, DeviceItem.

`<InternalLink>` tags name is unique across the common parent.

Attributes of an "IO-system" element

The following table shows the related attributes of the object for CAx import and export files:

Attribute	Handling	Comment
Name	Mandatory	The IO-system name. If empty string is imported, the IO-system is created with the default name.
Number	Optional	If not specified for import, the default value is applied.

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

8.6.28 Export/Import of multilingual comments

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)
- PLC is offline.

Application

The CAx data exchange exports and imports comments and multilingual comments of the following hardware objects:

- Devices (Device)
- Modules (DeviceItems)
- Tags (Tag)

The import/export of multilingual comments comprises all TIA Portal languages.

Restrictions

- Export
 - Only if a comment exists, a "Comment" attribute is exported to the AML file.
- Import
 - The "Comment" attribute is optional.
 - For virtual device items, no comments can be imported.

Example: Exported configuration with multilingual comments

The following figure shows the configuration of a SIMATIC S7 1500 (Device) with PLC_1 (DeviceItems). For both objects, comments are set in English, French, German and Chinese.

English (United States)	French (France)	German (Germany)	Chinese (People's Republic of China)	Reference
Profinet_Module	Profinet_Module_fr	Profinet_Module_de	Profinet_Module_chs	PROFINET interf...
Device_01	machine_01	Gerät_01	Device_01_chs	S7-1200 statio...
				Project2\PLC_1...
				Project2\PLC_1...

AML structure

After the export of this configurations, the multilingual comments are generated as nested attributes of the device, device item or tag.

- The parent attribute "Comment" shall have the value used in default language.
- A child attribute exists for every foreign-language comment.

```
...
<Attribute Name="Comment" AttributeDataType="xs:string">
  <Value>English</Value>
  <Attribute Name="aml-lang=en-US" AttributeDataType="xs:string">
    <Value>English</Value>
  </Attribute>
  <Attribute Name="aml-lang=de-DE" AttributeDataType="xs:string">
    <Value>Deutsch</Value>
  </Attribute>
  <Attribute Name="aml-lang=zh-HK" AttributeDataType="xs:string">
    <Value>Chinese</Value>
  </Attribute>
  ...
  ...
</Attribute>
...
```

See also

- [Structure of the CAx data for importing/exporting \(Page 893\)](#)
- [Connecting to the TIA Portal \(Page 76\)](#)
- [Opening a project \(Page 106\)](#)

8.6.29 AML AR APC 1.1 Attribute Support

8.6.29.1 Export/Import of PLC tags

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open.
See [Opening a project \(Page 106\)](#)
- PLC is offline.

Application

Exported and imported symbols and tags are assigned to a device item. CAx import/export concerns hardware oriented symbols and tags. The symbols and tags are exported only with the controller target device item, e. g. the CPU and not with other device items they might refer to, e. g. an I/O module. Like devices, the tags are often grouped in tag tables and in a hierarchical folder structure.

AML structure elements

PLC tags, tag tables and tag user folders can be exported and imported via CAx import/export function. The tag objects are mapped in the following AML structure elements:

- <InternalElement>
Tag tables and tag user folders are mapped as internal elements of the related PLC with the respective role class.
- <ExternalInterface>
Represents a PLC tag, dedicated to the internal element of the related tag table or tag user folder.

A mapping channel with a PLC tag is exported as communication partner via the <internal link> element. The following XML structure shows an example:

```
...
<InternalLink Name="Link To Tag_1"
  RefPartnerSideA="b33451f6-d88f-4900-8dbe-41f1be1e3535:Channel_DI_0"
  RefPartnerSideB="b2b937ee-d5db-4826-9340-027b1da22828:Tag_1" />
...
...
```

PLC tag user folder

The objects "TagUserFolder" only need the "Name" attribute in CAx import and export files.

Attributes of a PLC tag table

The following table shows the related attributes of the object for CAx import and export files:

Attribute	Handling	Comment
Name	Mandatory, ignored if "AssignToDefault" = TRUE	
AssignToDefault	Import-only	Used to identify the default tag table during import. If "AssignToDefault" = TRUE, all tags are created under the default tag table of the TIA Portal.

Attributes of a PLC tag

The following table shows the related attributes of the object for CAx import and export files:

Attribute	Handling	Comment
Name	Mandatory	
DataType	Mandatory	
LogicalAddress	Optional	Imported and exported in international mnemonics format
IoType	Optional	Input or output
Comment	Optional	

Note

TIA Portal V16 supports export and import of IoType attribute in AML file with AR APC V1.1.0

Example: AML structure

The following figure shows the structure of the following exported tag objects:

- empty default tag table
- tag user folder "Group_1"
- included tag table "Tag table_1"
- four tags

```
<InternalElement ID="a310b193-ba04-49d7-a8e3-004619c7d9d2" Name="Default tag table">
  <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/TagTable" />
</InternalElement>
<InternalElement ID="0feff703-9c70-4ca9-b3b3-8de8229696dd" Name="Group_1">
  <InternalElement ID="f92g9ce4-c015-459f-9f59-8f94bca3b186" Name="Tag table_1">
    <ExternalInterface ID="fc0c8c5a-fd5b-443b-b430-6435b6aa22ff" Name="Tag_1" RefBaseClassPath="AutomationProjectConfigurationI...
    ...
    </ExternalInterface>
    <ExternalInterface ID="450d6a1d-81b8-49ae-a104-c0072933d669" Name="Tag_2" RefBaseClassPath="AutomationProjectConfigurationI...
    ...
    </ExternalInterface>
    <ExternalInterface ID="3de17a36-b5c5-4fc7-9fc3-47e4a8f95087" Name="Tag_3" RefBaseClassPath="AutomationProjectConfigurationI...
      <Attribute Name="DataType" Attribute DataType="xs:string">
        <Value>Word</Value>
      </Attribute>
      <Attribute Name="IoType" Attribute DataType="xs:string">
        <Value>Input</Value>
      </Attribute>
      <Attribute Name="LogicalAddress" Attribute DataType="xs:string">
        <Value>IWO</Value>
      </Attribute>
    </ExternalInterface>
    <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/TagTable" />
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/TagUserFolder" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/TagUserFolder" />
</InternalElement>
...

```

See also

[Structure of the CAx data for importing/exporting \(Page 893\)](#)

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

8.6.29.2 Export/Import of RH/PLC

Requirement

- The TIA Portal Openness is connected to the TIA Portal
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- PLC is offline

Application

You can use TIA Portal to export and import AML file following AR APC V1.1 with R/H PLC having same IP Address configuration.

Attribute

TIA Portal supports only one attribute in AML file for R/H PLC, if available in TIA Portal user interface:

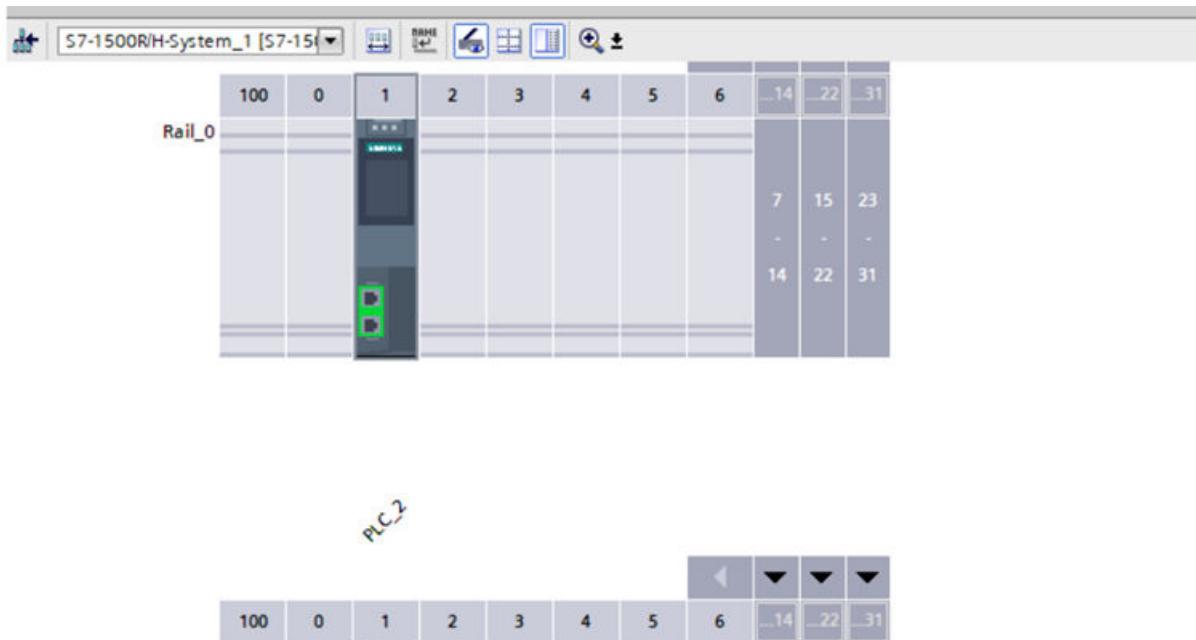
Device Item Attribute Name	Handling	Comment
HNetworkAddress	Mandatory	<p>Exported/Imported, if the device item R/H PLC supports this attribute in TIA Portal and is non empty otherwise it shall be skipped.</p> <p>For import, the "Enable the system IP address for switched connection" shall be set and the HNetworkAddress shall be valued.</p>

Restriction

- Export
 - Only if the "Enable the system IP address for switched connection" in TIA Portal is selected.

Example: Exported configuration

The following configuration shows a device item where the HNetworkAddress are configured.



AML structure of export file

The following figure shows the structure of exported AML file for R/H PLC

```

<InternalElement ID="e1879cf8-8222-4ce3-b171-60c56aed7f18" Name="PROFINET interface_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X1</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>32768</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<InternalElement ID="a32c67c7-47dc-4362-a9c4-b41b93b58eff" Name="E1">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>Ethernet</Value>
</Attribute>
<Attribute Name="NetworkAddress" AttributeDataType="xs:string">
<Value>192.168.0.1</Value>
</Attribute>
<Attribute Name="SubnetMask" AttributeDataType="xs:string">
<Value>255.255.255.0</Value>
</Attribute>
<Attribute Name="IpProtocolSelection" AttributeDataType="xs:string">
<Value>Project</Value>
</Attribute>
<Attribute Name="HNetworkAddress" AttributeDataType="xs:string">
<Value>192.168.0.3</Value>
</Attribute>
<ExternalInterface ID="802676fa-212f-4bf5-b112-de94993a0340" Name="LogicalEndPoint_Node">
RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Node" />
</InternalElement>

```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

8.6.29.3 Export/Import AML with customized tag and deviceitems

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open
See [Opening a project \(Page 106\)](#)
- PLC is offline

Application

You can use TIA Portal to export and import AML file following AR APC V1.1 with 'Customized' sub-attribute of tag and deviceitem.

Attribute

The following table shows the related attribute available for a tag during CAx import and export files:

Attribute Name	Handling	Comment
Customized	Optional for export/import	<p>Sub-attribute of 'DataType' of a tag.</p> <p>True and False are the only allowed values for Customized.</p> <p>While importing, the attribute customized has no relevance</p> <p>During export Customized sub-attribute has a value "true" for datatypes other than IEC 61131 datatypes. For IEC 61131 compliant datatypes the Customized sub-attribute shall not be exported.</p>

The following table shows the related attribute available for a deviceitem during CAx import and export files:

Attribute Name	Handling	Comment
Customized	Optional for import	<p>It is a child attribute to parent 'DeviceItemType' attribute for a DeviceItem.</p> <p>True and False are the only allowed values for Customized.</p> <p>While importing and exporting, Customized attribute has no relevance</p>

Exported AML file

The below AML file shall be generated during export of an AML file with customized deviceItems.

```
<?xml version="1.0" encoding="utf-8"?>
<CAEXFile FileName="Project26.aml" SchemaVersion="2.15"
xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd">
<AdditionalInformation>
<WriterHeader>
...
</WriterHeader>
</AdditionalInformation>
<AdditionalInformation AutomationMLVersion="2.0" />
<AdditionalInformation DocumentVersions="Recommendations">
<Document DocumentIdentifier="AR APC" Version="1.1.0" />
</AdditionalInformation>
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
<InternalElement ID="03ecf798-3e07-4976-b281-f8b98eb3a590" Name="Project26">
...
<InternalElement ID="c218aea9-93b8-4719-be6f-ccfa9517e2c6" Name="S71500/ET200MP station_1">
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>System:Device.S71500</Value>
</Attribute>
<InternalElement ID="c1ae306f-183f-47b8-91e6-cb331c559278" Name="Rail_0">
...
<InternalElement ID="8d7d5ee1-603a-4897-b47e-8954d4c21a31" Name="PLC_1">
...
<Attribute Name="DeviceItemType" AttributeDataType="xs:string">
<Value>CPU</Value>
</Attribute>
...
...
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
Device" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
AutomationProject" />
</InternalElement>
</InstanceHierarchy>
</CAEXFile>
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

8.6.29.4 Export/Import of FailSafe PLC**Requirement**

- The TIA Portal Openness application is connected to the TIA Portal.
Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- PLC is offline

Application

You can use TIA Portal to export and import an AML file following AR APC V1.1 with FailSafe PLC having failsafe attributes.

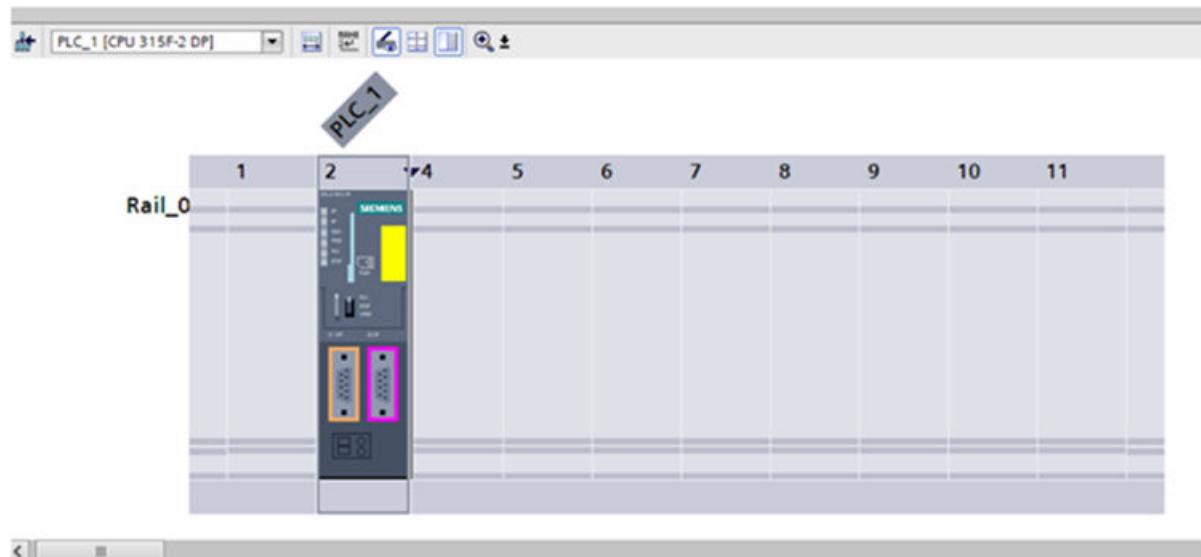
Attributes

The following table shows the list of failsafe attributes for CAx import and export AML files:

Openness Attribute	Handling	Comment	AR APC Name in AML
Failsafe_FSourceAddress	Mandatory	Export/import only if the device item is a failsafe PLC and supports this attribute in TIA Portal and is non empty otherwise it shall be skipped	Failsafe_FSourceAddress
Failsafe_LowerBoundForFDestinationAddresses	Mandatory	Exported/Imported only if the device item is a failsafe PLC and supports this attribute in TIA Portal and is non empty otherwise it shall be skipped	Failsafe_LowerBoundForFDestinationAddresses
Failsafe_UpperBoundForFDestinationAddresses	Mandatory	Exported/Imported only if the device item is a failsafe PLC and supports this attribute in TIA Portal and is non empty otherwise it shall be skipped	Failsafe_UpperBoundForFDestinationAddresses
Failsafe_CentralFSourceAddress	Optional	Exported/Imported only if the device item is a failsafe PLC and supports this attribute in TIA Portal and is non empty otherwise it shall be skipped	Failsafe_FSourceAddress
Failsafe_FDestinationAddress	Optional	Exported/Imported only if the device item is a failsafe PLC and supports this attribute in TIA Portal and is non empty otherwise it shall be skipped	Failsafe_FDestinationAddress

Example: Exported configuration

The following configuration shows a device item where the attributes are configured.



AML structure of the export file

The following figures show the structure of the exported AML file.

```
<InternalElement ID="9c944d2c-e0ae-4f39-b35a-a63faaf35be7" Name="PLC_1">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>CPU 1511TF-1 PN</Value>
</Attribute>
<Attribute Name="DeviceItemType" AttributeDataType="xs:string">
<Value>CPU</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 511-1UK01-0AB0</Value>
</Attribute>
<Attribute Name="InstallationDate" AttributeDataType="xs:dateTime">
<Value>2019-02-28T08:12:12.987Z</Value>
</Attribute>
<Attribute Name="Failsafe_FSourceAddress" AttributeDataType="xs:string">
<Value>1</Value>
</Attribute>
<Attribute Name="Failsafe_LowerBoundForFDestinationAddresses" AttributeDataType="xs:string">
<Value>1</Value>
</Attribute>
<Attribute Name="Failsafe_UpperBoundForFDestinationAddresses" AttributeDataType="xs:string">
<Value>99</Value>
</Attribute>
<Attribute Name="FirmwareVersion" AttributeDataType="xs:string">
<Value>V2.8</Value>
</Attribute>
<InternalElement ID="4f718e93-b541-4983-8f13-1f5b21c3e70c" Name="Default tag table">
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/TagTable"/>
</InternalElement>
<InternalElement ID="20e19f5b-8ace-4e0c-af0b-c710ae4817da" Name="CPU display_1">
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>3</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/DeviceItem" />
</InternalElement>
<InternalElement ID="5a24516f-17d6-4b2a-a4ac-efc1b577875d" Name="Card reader/writer_1">
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>4</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
```

8.6 Importing/exporting hardware data

```
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement> <InternalElement ID="0f746d71-035e-4e64-b0d7-51d0449cf88" Name="OPC
UA_1">
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>254</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value> </Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<InternalElement ID="a0633104-a2ac-4680-bb99-81df50f5ec40" Name="PROFINET interface_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X1</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>32768</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<InternalElement ID="e3497176-dbba-4fec-9d3a-772ae13987c4" Name="E1">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>Ethernet</Value> </Attribute>
<Attribute Name="NetworkAddress" AttributeDataType="xs:string">
<Value>192.168.0.1</Value>
</Attribute>
<Attribute Name="SubnetMask" AttributeDataType="xs:string">
<Value>255.255.255.0</Value>
</Attribute>
<Attribute Name="IpProtocolSelection" AttributeDataType="xs:string">
<Value>Project</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Node" />
</InternalElement>
<InternalElement ID="3208384f-d5ba-4ccb-b8da-f08ec38ec681" Name="Port_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>P1 R</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>32769</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<InternalElement ID="4a47c05e-9656-4e02-9b51-23b065b6fe47" Name="Port_2">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>P2 R</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>32770</Value>
</Attribute>
```

```
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<InternalElement ID="e3fdb611-4b68-4682-b154-ae43c74a24d3" Name="F-DI 16x24V DC_1">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>F-DI 16x24V DC</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>2</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value> </Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 526-1BH00-0AB0</Value>
</Attribute>
<Attribute Name="FirmwareVersion" AttributeDataType="xs:string">
<Value>V1.0</Value>
</Attribute>
<InternalElement ID="77c4fea0-baba-44e6-80f2-72b7b830a88a" Name="F-DI 16x24V DC_1">
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute> <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<Attribute Name="Failsafe_FSourceAddress" AttributeDataType="xs:string">
<Value>1</Value>
</Attribute>
<Attribute Name="Failsafe_FDestinationAddress" AttributeDataType="xs:string">
<Value>655</Value>
</Attribute>
</InternalElement>
</InternalElement>
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

8.6.29.5 Export/Import of Failsafe IO**Requirement**

- The TIA Portal Openness application is connected to the TIA Portal.
See Connecting to the TIA Portal (Page 76)
- A project is open
See Opening a project (Page 106)
- PLC is offline

Application

You can use TIA Portal to export and import an AML file following AR APC V1.1 with FailSafe IO having failsafe attributes.

Attributes

The following table shows the related attributes available on Failsafe IO module/submodule for CAx import and export files :

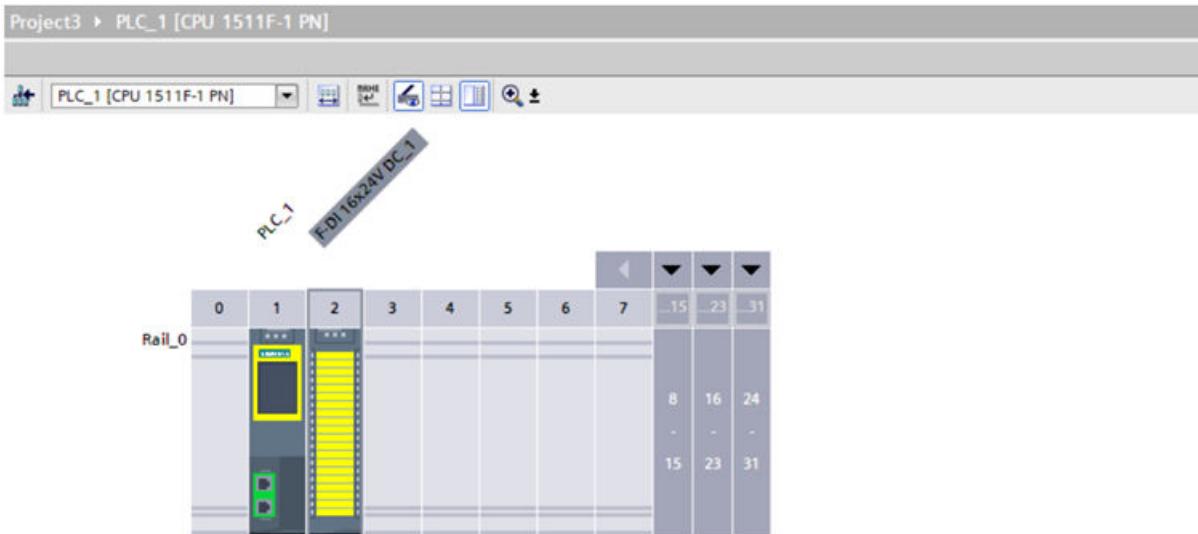
Device Item Attribute Name	Handling	Comment
FSourceAddress	Mandatory	Export/import only if the device item is a failsafe IO and supports in TIA Portal.
FDestinationAddresses	Mandatory	Export/import only if the device item is a failsafe IO and supports in TIA Portal.

Note

- For most of the Failsafe IO's, FSourceAddress is a ReadOnly it is not writable and is a reflection of FCentralFSourceAddress attribute of failsafe PLC. The same value shall get exported and ignored during import.
- If the FSourceAddress and FDestinationAddress are Readonly, then value shall be exported. However, during import warning message should be displayed in info tab of TIA Portal.
- FSourceAddress and FDestinationAddress should be supported for IO module and sub module level.

Example: Exported configuration

The following configuration shows a device item where the attributes are configured.



AML structure of the export file

The following figure shows the structure of exported AML file.

```
<Attribute Name="FSourceAddress" AttributeDataType="xs:string">
<Value>1</Value>
</Attribute>
<Attribute Name="FDestinationAddress" AttributeDataType="xs:string">
<Value>65534</Value>
</Attribute>
```

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

8.6.29.6 Export/Import vendor specific attribute

Requirement

- The TIA Portal Openness application is connected to the TIA Portal.
See [Connecting to the TIA Portal \(Page 76\)](#)
- A project is open
See [Opening a Project \(Page 106\)](#)

Application

In TIA Portal, you can export and import an AML file for device and deviceitems having "Manufacturer" attribute so that you can exchange vendor specific information in round trip scenarios.

The attribute "Manufacturer" is not supported by Openness. It shall be consumed through "Transformation Plugin" during Export/Import.

Note

AML files shall be generated(exported) by TIA Portal V16 following AR APC V1.1.

Export

You can export the "Manufacturer" attribute into AML file, if it is available in AmiModel, which shall be filled by "Transformation Plugin" users.

Import

You can import "Manufacturer" attribute to TIA Portal, wherein vendor specific attribute is available in AmiModel which shall be consumed by "Transformation Plugin" users.

Currently this attribute shall be supported only for 'Drive' devices and device items since only they do have official "Transformation Plugin" written for CAx.

See also

[Connecting to the TIA Portal \(Page 76\)](#)

[Opening a project \(Page 106\)](#)

8.6.30 AML attributes versus TIA Portal Openness attributes

Access attributes and export/import attributes

Via TIA Portal Openness you can access attributes of hardware objects. Single names you use to access these attributes e. g. of a device item differs from the attributes names in the export/import AML file.

Attributes list

The following table provides an overview to both kinds of attributes:

Table 8-6 Attribute names of devices and GSD/GSDML devices

AML file	TIA Portal Openness
Name	Name
TypeIdentifier	TypeIdentifier
Comment	Comment

Table 8-7 Attribute names of device items

AML file	TIA Portal Openness
Name	Name
TypeIdentifier	Mapped to substring of <TypeIdentifier> (i.e., value before first "/" operator) ignoring firmware version part in it. Mapping substring is applicable only when TypeIdentifier starts with <OrderNumber:> prefix and it has firmware version part otherwise mapped to complete <TypeIdentifier>.
FirmwareVersion	<FirmwareVersion> mapped to substring of <TypeIdentifier> (i.e., value after first "/" operator). Mapping substring is applicable only when <TypeIdentifier> starts with <OrderNumber:> prefix and it has firmware version part.
TypeName	TypeName
DeviceItemType (for CPU and HeadModule)	Classification
PositionNumber	PositionNumber
BuiltIn	IsBuiltIn
PlantDesignation IEC	PlantDesignation
LocationIdentifier IEC	LocationIdentifier
Comment	Comment

Table 8-8 Attribute names of GSD/GSDML device items

AML file	TIA Portal Openness
Name	Name
TypeIdentifier	TypeIdentifier
TypeName	TypeName
DeviceItemType (for HeadModule)	Classification
PositionNumber	PositionNumber
BuiltIn	IsBuiltIn
Comment	Comment
Label	Label

8.6 Importing/exporting hardware data

Table 8-9 Attribute names of tags

AML file	TIA Portal Openness
Name	Name
DataType	DataTypeName
LogicalAddress	LogicalAddress
Comment	Comment

Table 8-10 Attribute names of tag tables

AML file	TIA Portal Openness
Name	Name
AssignToDefault	IsDefault

Table 8-11 Attribute names of addresses

AML file	TIA Portal Openness
StartAddress	StartAddress
Length	Length
IoType	IoType

Table 8-12 Attribute names of ports

AML file	TIA Portal Openness
Name	Name
TypeIdentifier	TypeIdentifier
FirmwareVersion	FirmwareVersion
TypeName	TypeName
PositionNumber	PositionNumber
BuiltIn	IsBuiltIn
Comment	Comment
Label	Label

Table 8-13 Attribute names of devices with IO-interface

AML file	TIA Portal Openness
Name	Name
TypeIdentifier	TypeIdentifier
FirmwareVersion	FirmwareVersion
TypeName	TypeName
DeviceItem Type (for CPU and HeadModule)	Classification
PositionNumber	PositionNumber

AML file	TIA Portal Openness
Builtin	IsBuiltIn
Label	Label
Comment	Comment

Table 8-14 Attribute names of channels

AML file	TIA Portal Openness
Type	Type
IoType	IoType
Number	Not mapped to any attribute in TIA Portal Openness.
Length	ChannelWidth

Major Changes

9.1 Major changes in TIA Portal Openness V15.1

Changes

If you have considered the hints concerning programming across versions and you do not upgrade your project to V15.1 your applications will run without any restrictions on any computer even if only a TIA Portal V15.1 is installed.

If you upgrade your project to V15.1 it is necessary to recompile your application using the SiemensEngineering.dll of V15.1. In some cases it might be necessary to adapt the code of your application

Type identifiers

The type identifier for racks and devices of "PC with ports" and "Ethernet devices with ports" have been renamed.

PC with ports	Type identifier before V15.1	Type identifier as of V15.1
Device	System:DesktopPC.Device	System:Device/DesktopPC
Rack	System:DesktopPC.Rack	System:Rack/DesktopPC
Device item	System:DesktopPC.Port<X> <X> denotes the number of ports	System:DeviceItem/EthernetDevice.Port<X> <X> denotes the number of ports

Ethernet devices with ports		
Device	System:DummyPC.Device	System:Device/EthernetDevice
Rack	System:Rack/DummyPC	System:Rack/EthernetDevice
Device item	System:DummyPC.Port<X> <X> denotes the number of ports	

Failures when trying to connect to TIA Portal

The messages in case of failures when trying to connect to TIA Portal have been enhanced in a more specific way.

Major Changes

9.1 Major changes in TIA Portal Openness V15.1

Cross-thread operations

Access to Openness objects is not inherently thread-safe.

If you use multi-threading to improve the performance of your Openness application, it is recommended to create your TIA Portal instance with an MTA.

If TIA Portal is created or attached within an STA thread, all Openness objects associated with that Portal instance must be accessed from the same STA thread; otherwise, an exception will be thrown.

Submodules do not have attributes Author and TypeName

The attributes Author and TypeName have been removed from submodules which cannot be plugged.

Opening of a global library

As of TIA Portal Openness V15.1 a global library can be opened via Openness independant of a persisted preview mode of the library.

Application exit codes

In case of an application exit code

- Up to TIA Portal Openness V15 you get a non recoverable exception
- As of TIA Portal Openness V15.1 you get a EngineeringRuntimeException or a EngineeringTargetException if the error code is known and a non recoverable exception if the error code is unknown.

Schema extension for nameless parameters

The import of SCL blocks is possible even if ENO is used at a nonformal call..

Header of indexed parameters

As of TIA Portal Openness V15.1 the header of indexed parameters my not be changed via Openness.

Some drive parameters are modelled as indexed parameters, as they provide several pieces of data for one topic. The modeling of the indexed drive parameters in Openness follows the drive specific definition of the respective parameters as they are defined in the respective list manual.

Indexed parameters are modeled as follows:

Header item: The respective drive parameter without any index. The header item contains a descriptive text, to inform you about the semantics of the referenced indexed drive parameter. Therefore the header item is readonly, because it holds no actual value.

Indexed Items: The indexed items of the drive parameter below the header item. They provide descriptive text, defining the semantics of the specific indexed item (readonly). In addition, they also provide the value that can be retrieved via Openness API. If the respective indexed parameter is also writeable, than the value can also be set via Openness API.

Attribute TransmissionRateAndDuplex

Some faulty enum values for attribute TransmissionRateAndDuplex have been corrected, for example the enum value "POFPCF100MbpsFullDuplexLD" has been removed and "POFPCF100MbpsFullDuplex" has been added. For detail information, please see Configurable attributes of a port-to-port connection (Page 133)

Attribute AutoNumber for know how protected blocks

As of V15.1 the attribute AutoNumber can not be changed via TIA Portal Openness if a block is know how protected.

Number of channels listed by ChannelInfo interface

Up to TIA Portal Openness V15 the ChannelInfo interface the number of available channels hasn't been correct for some modules.

Access to ProDiag FB attributes

The following attributes of a ProDiag FB can be accessed via TIA Portal Openness:

- Version
- Initial values acquisition
- Use central timestamp

Import/Export of failsafe blocks

Import of failsafe blocks from previous versions is not possible.

Export of system generated failsafe blocks will be prevented as of TIA Portal Openness V15.1.

R/H systems

Download or access for R/H devices is available at device.

Online and Download Provider are not available for individual R/H PLC (DeviceItems).

For PLC2 of an R/H system, SoftwareContainer will not be available.

9.2 Major changes in TIA Portal Openness V15

Changes

If you have considered the hints concerning programming across versions and you do not upgrade your project to V15 your applications will run without any restrictions on any computer even if only a TIA Portal V15 is installed.

If you upgrade your project to V15 it is necessary to recompile your application using the SiemensEngineering.dll of V15.

In some cases it is necessary to adapt the code of your application

- Behaviour changes for compositions in DeviceItemComposition
- BitOffset of ASI addresses
- Exception class
- System folders of system UDTs
- Submodules do not have attributes Author and TypeName
- Timestamp for last modification
- Export XML for GRAPH blocks
- Importing tag tables
- Modifying not failsafe relevant attributes of a PLC
- Modifying F-parameters while safety password is set
- Accessing TO objects in a S7 1200 CPU

Behaviour changes for compositions in DeviceItemComposition

The following compositions in DeviceItemCompositon have been changed for a dynamic behaviour. The composition is updated now if an element is added or deleted via the user interface of TIA Portal.

- IoSystem - ConnectedIoDevices
- Subnet - IoSystems
- Subnet - Nodes
- NetworkInterface - Nodes
- NetworkInterface - Ports
- NetworkPort - ConnectedPorts
- SubnetOwner - Subnets

BitOffset of ASI addresses

If a module has an input and an output address for both address objects the correct attribute BitOffset will be provided.

If a module has channels the attribute BitOffset will not be provided for the channel.

Exception class

ServiceID and MessageID have been removed from exception class

Submodules do not have attributes Author and TypeName

The attributes Author and TypeName have been removed from submodules which cannot be plugged.

System folders of system UDTs

For system folders of system UDTs the appropriate folder and composition is provided. This leads also to a change in the hierarchy of compare results.

Timestamp for last modification

If during an upgrade an object is changed the timestamp for the last modification is changed as well.

Export XML for GRAPH blocks

The export XML for GRAPH blocks contains an additional empty action: <Actions />

Importing tag tables

Setting tag attributes is not longer dependent from data types.

Modifying not failsafe relevant attributes of a PLC

All not failsafe relevant attributes of a PLC can be modified via TIA Portal Openness, even if a safety password is set.

Modifying F-parameters while safety password is set

F-parameters of a F-IO can only be modified if the safety password is not set.

Accessing TO objects in a S7 1200 CPU

The access to array tags for the TO objects TO_PositioningAxis and TO_CommandTable has been changed. You can find details in the chapter about S7-1200 Motion Control.

9.3 Major changes in V14 SP1

9.3.1 Major changes in V14 SP1

Introduction

The following changes were made in TIA Portal Openness API object modell V14 SP1, which may impact your existing applications:

Change	Required program code adjustment
Improved handling for master copies	<p>The CreateFrom action will create a new object based on a master copy in a library and place it in the composition where the action was called. The CreateFrom action only supports master copies containing only single objects. The return type corresponds to the respective composed type.</p> <p>The following composition support CreateFrom:</p> <ul style="list-style-type: none">• Siemens.Engineering.HW.DeviceComposition• Siemens.Engineering.HW.DeviceItemComposition• Siemens.Engineering.SW.Blocks.PlcBlockComposition• Siemens.Engineering.SW.Tags.PlcTagTableComposition• Siemens.Engineering.SW.Tags.PlcTagComposition• Siemens.Engineering.SW.Types.PlcTypeComposition• Siemens.Engineering.SW.TecnologicalObjects.TecnologicalInstanceDBComposition• Siemens.Engineering.SW.Tags.PlcUserConstantComposition• Siemens.Engineering.Hmi.Tag.TagTableComposition• Siemens.Engineering.Hmi.Tag.TagComposition• Siemens.Engineering.Hmi.Screen.ScreenComposition• Siemens.Engineering.Hmi.Screen.ScreenTemplateComposition• Siemens.Engineering.Hmi.RuntimeScripting.VBScriptComposition• Siemens.Engineering.HW.SubnetComposition• Siemens.Engineering.HW.DeviceUserGroupComposition• Siemens.Engineering.SW.Blocks.PlcBlockUserGroupComposition• Siemens.Engineering.SW.ExternalSources.PlcExternalSourceUserGroupComposition• Siemens.Engineering.SW.Tags.PlcTagTableUserGroupComposition• Siemens.Engineering.SW.Types.PlcTypeUserGroupComposition
Improved handling for global libraries	<p>Existing actions on global libraries can now be modifying actions, e.g. delete a master copy from a global library.</p> <p>UpdateProject and UpdateLibrary do not longer use the UpdatePathsMode and DeleteUnusedVersionsMode parameters. Unused versions are not deleted after an update</p>

Change	Required program code adjustment
Change System.String to System.IO.FileInfo Change System.String to System.IO.DirectoryInfo	All occurrences where a string path had to be specified are using FileInfo path or a DirectoryInfo path. For example: <ul style="list-style-type: none">• Open project• Open library• Create project• Create global library• ...

New items in the object model

Name	Type	Namespace	Comment
PlcUserConstant	Class	Siemens.Engineering.SW.Tags	Split from PlcConstant.
PlcUserConstantComposition	Class	Siemens.Engineering.SW.Tags	Split from PlcConstantComposition.
PlcSystemConstant	Class	Siemens.Engineering.SW.Tags	Split from PlcConstant.
PlcSystemConstantComposition	Class	Siemens.Engineering.SW.Tags	Split from PlcConstantComposition.
MultilingualTextItem	Class	Siemens.Engineering	Access to multilingual text.
MultilingualTextItemComposition	Class	Siemens.Engineering	Access to multilingual text.
TiaPortalTrustAuthority.FeatureTokens	Enum value	Siemens.Engineering	Access to TIA Portal settings.
TiaPortalSetting	Class	Siemens.Engineering.Settings	Access to TIA Portal settings.
TiaPortalSettingComposition	Class	Siemens.Engineering.Settings	Access to TIA Portal settings.
TiaPortalSettingsFolder	Class	Siemens.Engineering.Settings	Access to TIA Portal settings.
TiaPortalSettingsFolderComposition	Class	Siemens.Engineering.Settings	Access to TIA Portal settings.
LanguageAssociation	Class	Siemens.Engineering	Access to active languages.
LanguageComposition.Find	Method	Siemens.Engineering	Access to active languages.

Modified items in the object model

Name	Type	Namespace	Comment
PlcConstant	Class	Siemens.Engineering.SW.Tags	Published base class of PlcUserConstant and PlcSystemConstant.
PlcTag	Class	Siemens.Engineering.SW.Tags	Split from PlcConstantComposition.
ITargetComparable	Interface	Siemens.Engineering.Compare	String attribute DataTypeName instead of an open link DataType.
MultilingualText	Class	Siemens.Engineering	Access to multilingual text.

Major Changes

9.3 Major changes in V14 SP1

Name	Type	Namespace	Comment
ProjectComposition.Create	Method	Siemens.Engineering	Parameters changed to using a DirectoryInfo and string.
Project.Subnets	Attribute	Siemens.Engineering	Access to subnets
Project.Languages	Attribute	Siemens.Engineering	Moved to be an attribute of Siemens.Engineering.LanguageSettings to provide supported languages

Removed items in the object model

Name	Type	Namespace	Comment
PlcConstantComposition	Class	Siemens.Engineering.SW.Tags	Split in PlcSystemConstantComposition and PlcUserConstantComposition.
CompareResultElement.PathInformation	Attribute	Siemens.Engineering.SW.Tags	Not used anymore.
MultilingualText.GetText(CultureInfo cultureInfo)	Method	Siemens.Engineering.Compare	Modified concept for accessing text items of MultilingualText.
TiaPortalTrustAuthority.CustomerIdentification	Enum value	Siemens.Engineering	Not used anymore.
TiaPortalTrustAuthority.ElevatedAccessExtensions	Enum value	Siemens.Engineering	Not used anymore.

Behaviour changes

Name	Type	Namespace	Comment
PlcTag.Export(FileInfo path, ExportOptions options)	Method	Siemens.Engineering.SW.Tags	The value of the attribute LogicalAddress is always exported in international mnemonics now. German mnemonics are still accepted on import.
PlcTag.LogicalAddress	Attribute	Siemens.Engineering.SW.Tags	The value of the attribute LogicalAddress is always returned in international mnemonics now. German mnemonics are accepted on write.

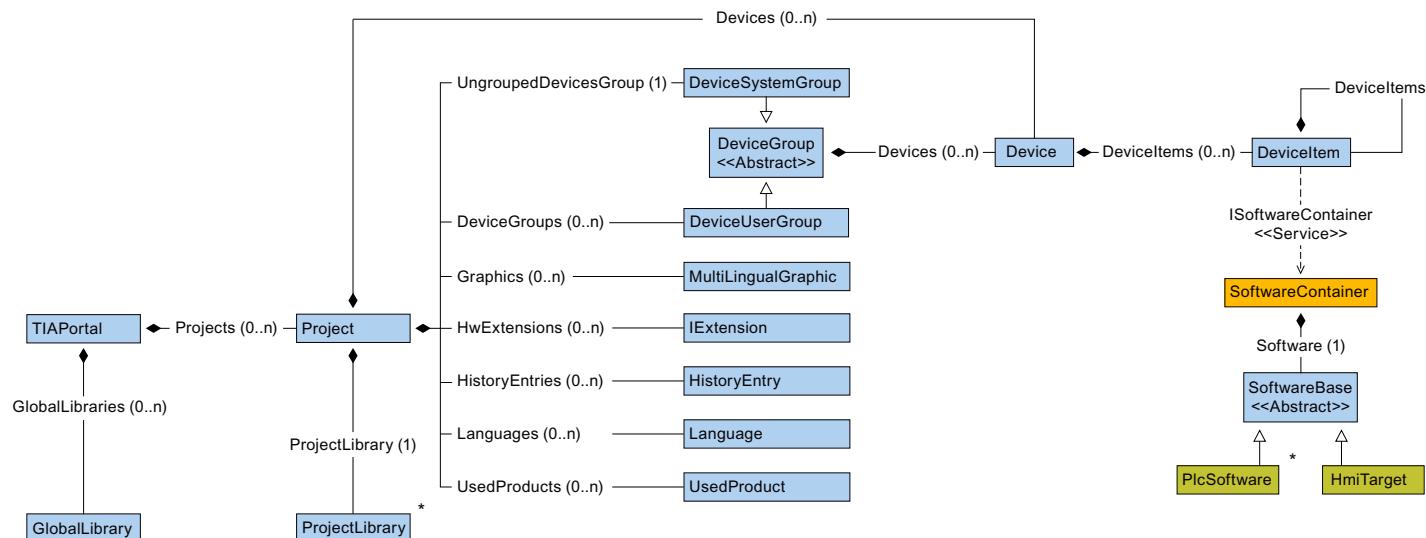
9.3.2 Major changes in the object model

Object model of TIA Portal Openness V14

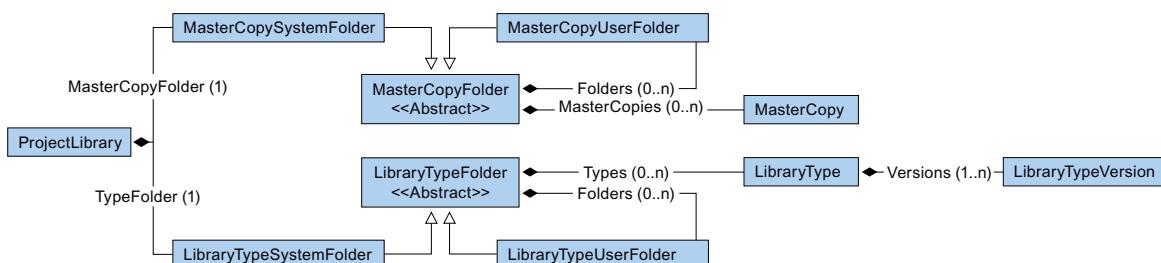
In order to allow you a comparison between the old and the new object model of TIA Portal Openness, the diagram below describes the object model of TIA Portal V14.

Note

The object model described on the diagram is obsolete, for information about the object model of TIA Portal Openness V14 SP1 refer to AUTOHOTSPOT



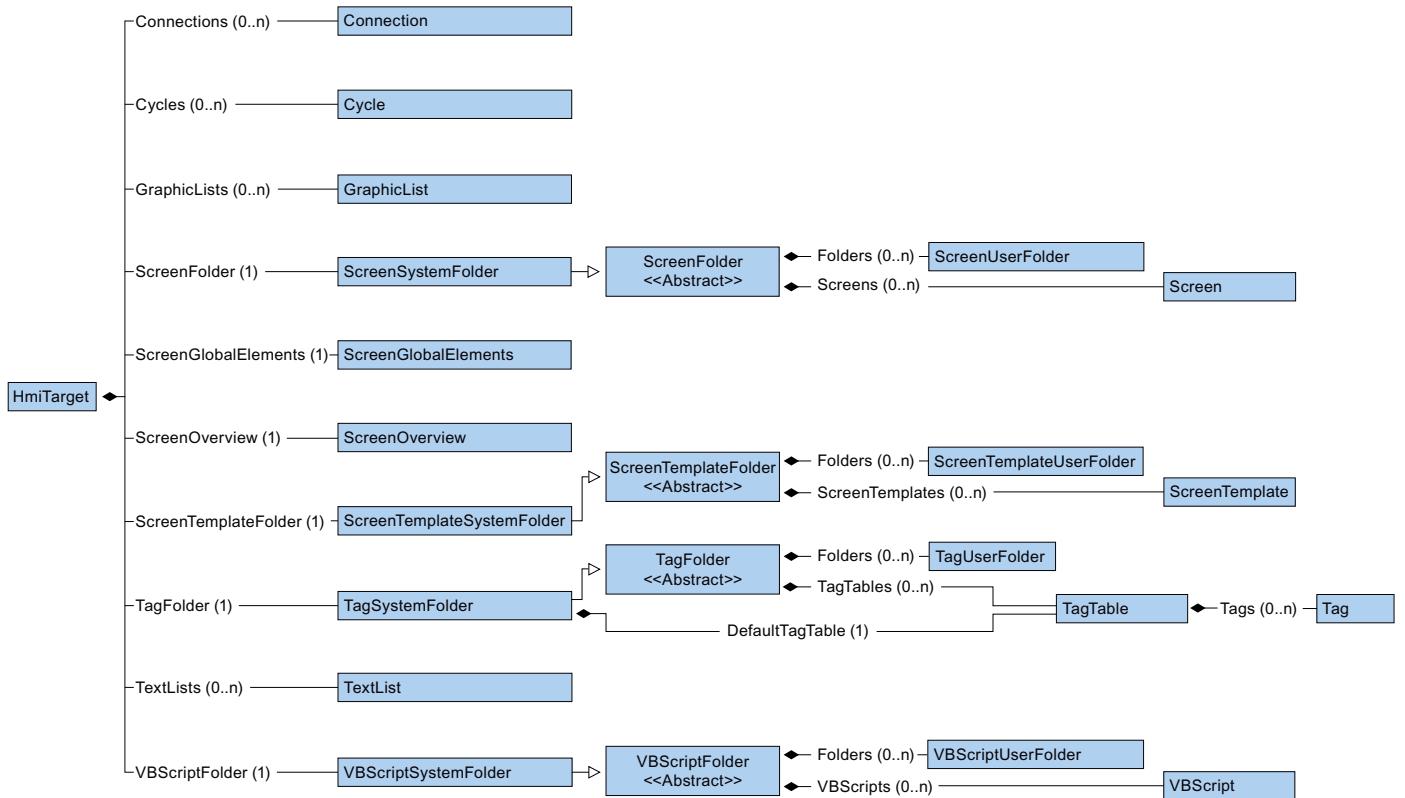
The following diagramm describes the objects which are located under ProjectLibrary.



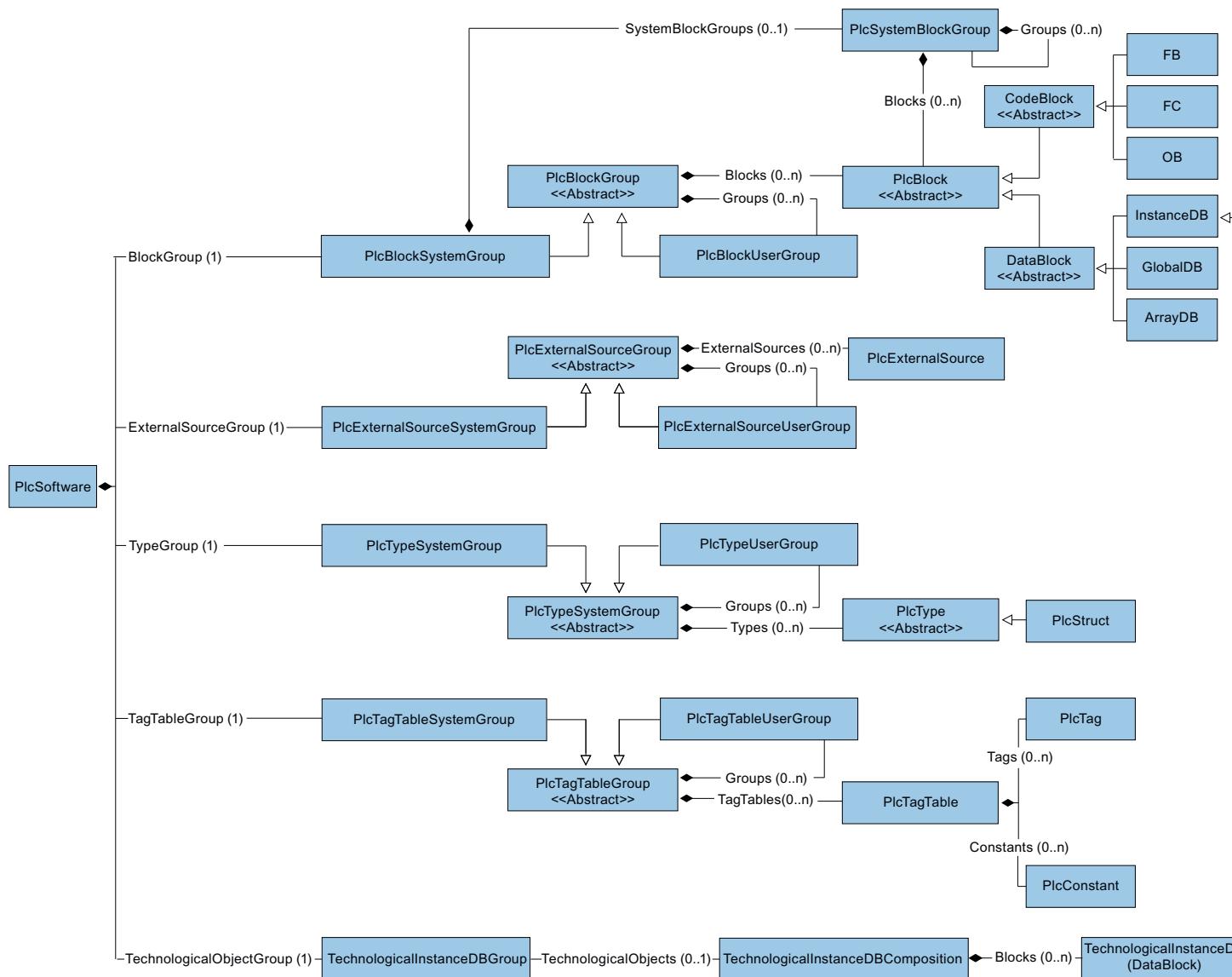
The following diagramm describes the objects which are located under HmiTarget.

Major Changes

9.3 Major changes in V14 SP1

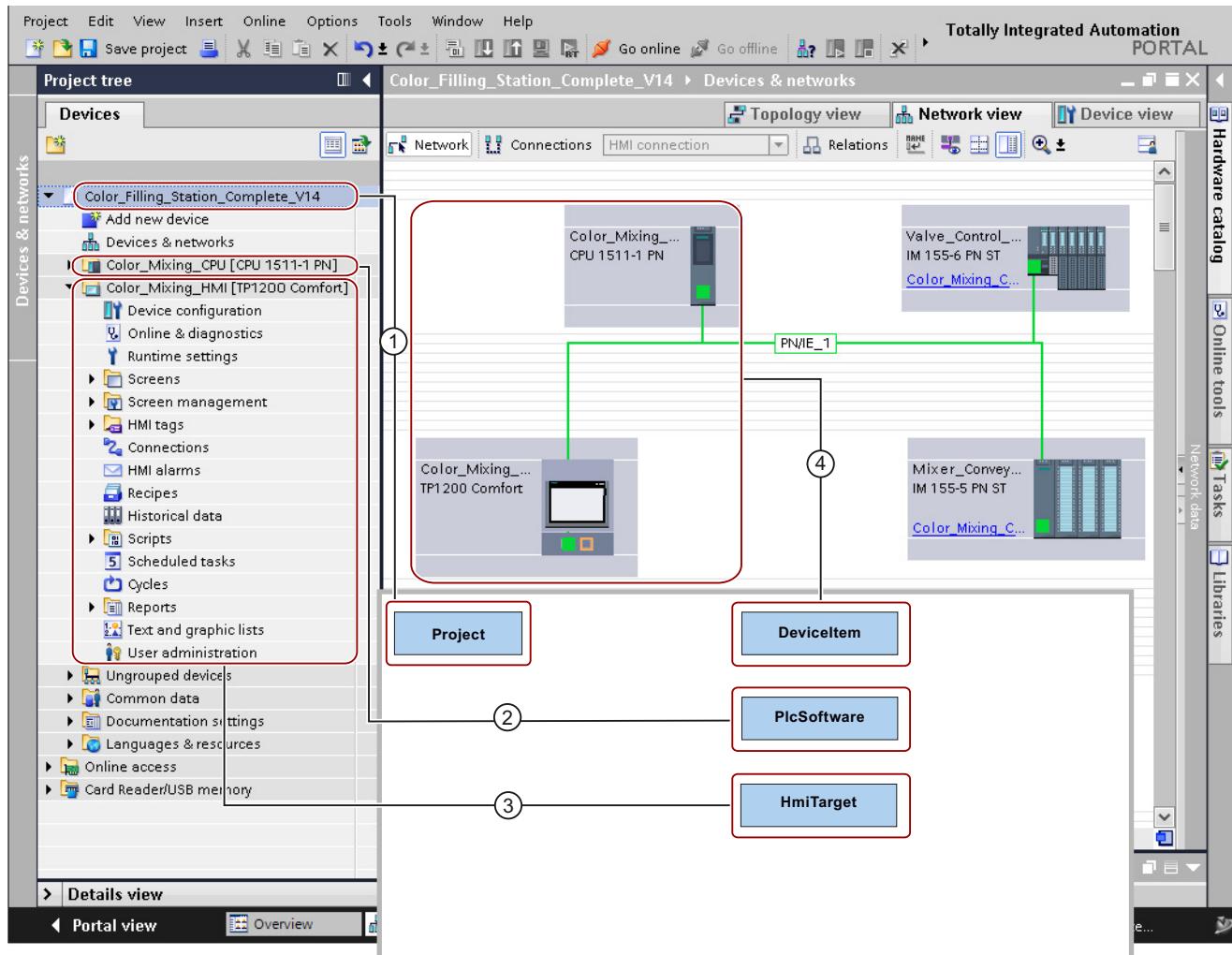


The following diagramm describes the objects which are located under PlcSoftware.



Relationship between TIA Portal and TIA Portal Openness object model

The figure below shows the relationship between the object model and a project in the TIA Portal:



- ① The object "Project" corresponds to an open project in the TIA Portal.
- ② The "PlcSoftware" object is of type "SoftwareBase"④, and corresponds to a PLC. The object's contents correspond to a PLC in the project navigation with access to objects such as blocks or PLC tags.
- ③ The "HmiTarget" object is of type "SoftwareBase"④, and corresponds to an HMI device. The object's contents correspond to an HMI device in the project navigation with access to objects such as screens or HMI tags.
- ④ The object "Deviceltem" corresponds to an object in the "Devices & Networks" editor. An object of the type "Deviceltem" can be a rack or an inserted module.

9.3.3 Changes on pilot functionality

Introduction

The following changes were made in API object modell V14 SP1 are only relevant for users, which have used the pilot functionality of HW Config in V14.

Modifications for TIA Portal Openness API types

TIA Portal Openness API type	new TIA Portal Openness API type
Siemens.Engineering.HW.IAddress	Siemens.Engineering.HW.Address
Siemens.Engineering.HW.IAddressController	Siemens.Engineering.HW.Features.AddressController
Siemens.Engineering.HW.IChannel	Siemens.Engineering.HW.Channel
Siemens.Engineering.HW.IDevice	Siemens.Engineering.HW.Device
Siemens.Engineering.HW.IDeviceItem	Siemens.Engineering.HW.DeviceItem
Siemens.Engineering.HW.IExtension	Siemens.Engineering.HW.Extensions
Siemens.Engineering.HW.IGsd	Siemens.Engineering.HW.Features.GsdObject
Siemens.Engineering.HW.IGsdDevice	Siemens.Engineering.HW.Features.GsdDevice
Siemens.Engineering.HW.IGsdDeviceItem	Siemens.Engineering.HW.Features.GsdDeviceItem
Siemens.Engineering.HW.IHardwareObject	Siemens.Engineering.HW.HardwareObject
Siemens.Engineering.HW.IHwIdentifier	Siemens.Engineering.HW.HwIdentifier
Siemens.Engineering.HW.IHwIdentifierController	Siemens.Engineering.HW.Features.HwIdentifierController
Siemens.Engineering.HW.IIoConnector	Siemens.Engineering.HW.IoConnector
Siemens.Engineering.HW.IIoController	Siemens.Engineering.HW.IoController
Siemens.Engineering.HW.IIoSystem	Siemens.Engineering.HW.IoSystem
Siemens.Engineering.HW.IInterface	Siemens.Engineering.HW.Features.NetworkInterface
Siemens.Engineering.HW.Extensions.ModuleInformationProvider	Siemens.Engineering.HW.Utilities.ModuleInformationProvider
Siemens.Engineering.HW.INode	Siemens.Engineering.HW.Node
Siemens.Engineering.HW.OPCUAExportProvider	Siemens.Engineering.HW.Utilities.OpcUaExportProvider
Siemens.Engineering.HW.IPort	Siemens.Engineering.HW.Features.NetworkPort
Siemens.Engineering.HW.IRole	Siemens.Engineering.HW.Features.HardwareFeature Siemens.Engineering.HW.Features.DeviceFeature Siemens.Engineering.HW.Utilities.ModuleInformationProvider
Siemens.Engineering.HW.SoftwareBase	Siemens.Engineering.HW.Software
Siemens.Engineering.HW.ISubnet	Siemens.Engineering.HW.Subnet
Siemens.Engineering.HW.ISoftwareContainer	Siemens.Engineering.HW.Features.SoftwareContainer
Siemens.Engineering.HW.ISubnetOwner	Siemens.Engineering.HW.Features.SubnetOwner

Major Changes

9.3 Major changes in V14 SP1

Modifications for Enums

TIA Portal Openness API type	Data type	new TIA Portal Openness API type	Data type
Siemens.Engineering.HW.Enums.AddressContext		Siemens.Engineering.HW.AddressContext	
Siemens.Engineering.HW.Enums.AddressIoType		Siemens.Engineering.HW.AddressIoType	
Siemens.Engineering.HW.Enums.Attachment-Type		Siemens.Engineering.HW.MediumAttachment-Type	
Siemens.Engineering.HW.Enums.BaudRate		Siemens.Engineering.HW.BaudRate	
Siemens.Engineering.HW.Enums.BusLoad		Siemens.Engineering.HW.CommunicationLoad	
Siemens.Engineering.HW.Enums.BusProfile		Siemens.Engineering.HW.BusProfile	
Siemens.Engineering.HW.Enums.CableLength		Siemens.Engineering.HW.CableLength	
Siemens.Engineering.HW.Enums.CableName	ulong	Siemens.Engineering.HW.CableName	long
Siemens.Engineering.HW.Enums.ChannelIoType	byte	Siemens.Engineering.HW.ChannelIoType	int
Siemens.Engineering.HW.Enums.ChannelType	byte	Siemens.Engineering.HW.ChannelType	int
Siemens.Engineering.HW.Enums.DeviceItem-Classifications		Siemens.Engineering.HW.DeviceItemClassifications	
Siemens.Engineering.HW.Enums.InterfaceOperatingModes		Siemens.Engineering.HW.InterfaceOperatingModes	
Siemens.Engineering.HW.Enums.IpProtocolSelection		Siemens.Engineering.HW.IpProtocolSelection	
Siemens.Engineering.HW.Enums.MediaRedundancyRole		Siemens.Engineering.HW.MediaRedundancyRole	
Siemens.Engineering.HW.Enums.NetType		Siemens.Engineering.HW.NetType	
Siemens.Engineering.HW.Enums.ProfinetUpdate-TimeMode		removed	
Siemens.Engineering.HW.Enums.RtClass	byte	Siemens.Engineering.HW.RtClass	int
Siemens.Engineering.HW.Enums.SignalDelaySelec-tion	byte	Siemens.Engineering.HW.SignalDelaySelection	int
Siemens.Engineering.HW.Enums.SyncRole	byte	Siemens.Engineering.HW.SyncRole	int
Siemens.Engineering.HW.Enums.Transmission-RateAndDuplex	uint	Siemens.Engineering.HW.TransmissionRateAnd-Duplex	int

Modifications for attributes values of Siemens.Engineering.HW.IoConnector

Attribut	Data type	new name	Data type
ProfinetUpdateTimeMode	ProfinetUpdateTime-Mode	PnUpdateTimeAutoCalculation	bool
ProfinetUpdateTime		PnUpdateTime	
AdaptUpdateTime		PnUpdateTimeAdaption	
WatchdogFactor		PNWatchdogFactor	
		DeviceNumber	string

Modifications for attributes values of Siemens.Engineering.HW.IoController

Attribut	Data type	new name	Data type
		DeviceNumber	string

Modifications for attributes values of Siemens.Engineering.HW.Node

Attribut	Data type	new name	Data type
HighestAddress		removed, available only on subnet	
TransmissionSpeed		removed, available only on subnet	
IsoProtocolUsed		UseIsoProtocol	
IpProtocolUsed		UseIpProtocol	
RouterAddressUsed		UseRouter	
PnDeviceNameAutoGenerated		PnDeviceNameAutoGeneration	
DeviceNumber		removed, moved to IoConnector / IoController	

Modifications for attributes values of Siemens.Engineering.HW.Subnet

Attribut	Data type	new name	Data type
HighestAddress	byte	HighestAddress	int
CableConfiguration		PbCableConfiguration	
RepeaterCount		PbRepeaterCount	
CopperCableLength		PbCopperCableLength	
OpticalComponentCount		PbOpticalComponentCount	
OpticalCableLength		PbOpticalCableLength	
OpticalRingEnabled		PbOpticalRing	
OlmP12		PbOlmP12	
OlmG12		PbOlmP12	
OlmG12Eec		PbOlmG12Eec	
OlmG121300		PbOlmG121300	
AdditionalNetworkDevices		PbAdditionalNetworkDevices	
AdditionalDpMaster	byte	PbAdditionalDpMaster	int
TotalDpMaster	byte	PbTotalDpMaster	int
AdditionalPassiveDevice	byte	PbAdditionalPassiveDevice	int
TotalPassiveDevice	byte	PbTotalPassiveDevice	int
AdditionalActiveDevice	byte	PbAdditionalActiveDevice	int
TotalActiveDevice	byte	PbTotalActiveDevice	int
PbCommunicationLoad	BusLoad	PbAdditionalCommunicationLoad	CommunicationLoad
OptimizeDde		PbDirectDateExchange	
MinimizeTslot		PbMinimizeTslotForSlaveFailure	
OptimizeCableConfig		PbOptimizeCableConfiguration	
CyclicDistribution		PbCyclicDistribution	
TslotInit		PbTslotInit	

Major Changes

9.3 Major changes in V14 SP1

Attribut	Data type	new name	Data type
Tslot		PbTslot	
MinTsdr		PbMinTsdr	
MaxTsdr		PbMaxTsdr	
Tid1		PbTid1	
Tid2		PbTid2	
Trdy		PbTrdy	
Tset		PbTset	
Tqui		PbTqui	
Ttr		PbTtr	
TtrMs		removed	
TtrTypical		PbTtrTypical	
TtrTypicalMs		removed	
Watchdog		PbWatchdog	
WatchdogMs		removed	
Gap	byte	PbGapFactor	int
RetryLimit	byte	PbRetryLimit	int
IsochronMode		IsochronousMode	
AdditionalDevice		PbAdditionalPassivDeviceForIsochronousMode	
TotalDevice		PbTotalPassivDeviceForIsochronousMode	
DpCycleTimeAutoCalc		DpCycleMinTimeAutoCalculation	
TiToAutoCalc		IsochronousTiToAutoCalculation	
Ti		IsochronousTi	
To		IsochronousTo	

Modifications for attributes values of Siemens.Engineering.Project

Attribut	Data type	new name	Data type
.HwExtensions		.HwUtilities	

Modifications for attributes values of Siemens.Engineering.HW.Baudrate

Attribut	Data type	new name	Data type
BaudRate.BAUD_9600		BaudRate.BAUD9600	
BaudRate.BAUD_19200		BaudRate.BAUD19200	
BaudRate.BAUD_45450		BaudRate.BAUD45450	
BaudRate.BAUD_93750		BaudRate.BAUD93750	
BaudRate.BAUD_187500		BaudRate.BAUD187500	
BaudRate.BAUD_500000		BaudRate.BAUD500000	
BaudRate.BAUD_1500000		BaudRate.BAUD1500000	
BaudRate.BAUD_3000000		BaudRate.BAUD3000000	

Attribut	Data type	new name	Data type
BaudRate.BAUD_6000000		BaudRate.BAUD6000000	
BaudRate.BAUD_12000000		BaudRate.BAUD12000000	

Modifications for attributes values of Siemens.Engineering.HW.CableLength

Attribut	Data type	new name	Data type
CableLength.Unknown		CableLength.None	
CableLength.Length_20m		CableLength.Length20m	
CableLength.Length_50m		CableLength.Length50m	
CableLength.Length_100m		CableLength.Length100m	
CableLength.Length_1000m		CableLength.Length1000m	
CableLength.Length_3000m		CableLength.Length3000m	

Modifications for attributes values of Siemens.Engineering.HW.ChannelloType

Attribut	Data type	new name	Data type
ChannelloType.Unknown		ChannelloType.Complex	

Modifications for attributes values of Siemens.Engineering.HW.IpProtocolSelection

Attribut	Data type	new name	Data type
IpProtocolSelection.AddressTailoring		IpProtocolSelection.VialoController	

Modifications for attributes values of Siemens.Engineering.HW.TransmissionRateAndDuplex

Attribut	Data type	new name	Data type
TransmissionRateAndDuplex.Unknown		TransmissionRateAndDuplex.None	
TransmissionRateAndDuplex.TP10Mbps_HalfDuplex		TransmissionRateAndDuplex.TP10MbpsHalfDuplex	
TransmissionRateAndDuplex.TP10Mbps_FullDuplex		TransmissionRateAndDuplex.TP10MbpsFullDuplex	
TransmissionRateAndDuplex.AsyncFiber10Mbps_HalfDuplex		TransmissionRateAndDuplex.AsyncFiber10MbpsHalfDuplex	
TransmissionRateAndDuplex.AsyncFiber10Mbps_FullDuplex		TransmissionRateAndDuplex.AsyncFiber10MbpsFullDuplex	
TransmissionRateAndDuplex.TP100Mbps_HalfDuplex		TransmissionRateAndDuplex.TP100MbpsHalfDuplex	
TransmissionRateAndDuplex.TP100Mbps_FullDuplex		TransmissionRateAndDuplex.TP100MbpsFullDuplex	
TransmissionRateAndDuplex.FO100Mbps_FullDuplex		TransmissionRateAndDuplex.FO100MbpsFullDuplex	

Major Changes

9.3 Major changes in V14 SP1

Attribut	Data type	new name	Data type
TransmissionRateAndDuplex.X1000Mbps_FullDuplex		TransmissionRateAndDuplex.X1000MbpsFullDuplex	
TransmissionRateAndDuplex.FO1000Mbps_Full-Duplex_LD		TransmissionRateAndDuplex.FO1000MbpsFull-DuplexLD	
TransmissionRateAndDuplex.FO1000Mbps_Full-Duplex		TransmissionRateAndDuplex.FO1000MbpsFull-Duplex	
TransmissionRateAndDuplex.TP1000Mbps_Full-Duplex		TransmissionRateAndDuplex.TP1000MbpsFull-Duplex	
TransmissionRateAndDuplex.FO10000Mbps_Full-Duplex		TransmissionRateAndDuplex.FO10000MbpsFull-Duplex	
TransmissionRateAndDuplex.FO100Mbps_Full-Duplex_LD		TransmissionRateAndDuplex.FO100MbpsFull-DuplexLD	
TransmissionRateAndDuplex.POFCF100Mbps_FullDuplex_LD		TransmissionRateAndDuplex.POFCF100MbpsFullDuplexLD	

9.3.4 Changes for export and import

9.3.4.1 Changes for export and import

Introduction

The export and import via TIA Portal Openness API was extended in V14 SP1 in order to handle comments at array elements. This required a new schema. Block interface import and export will handle from now on two schema versions:

- For import: The decision about used schema version is made based on namespace:
<Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
- For export: The decision about used schema version is made based on the project version.
Projects V14 SP1 lead to version 2, projects V14 lead to version v1

9.3.4.2 Changes in API

Generate source

The following methods have been removed from ProgramBlocks:

- GenerateSourceFromBlocks
- GenerateSourceFromTypes

The following methods have been added:

- GenerateSource to PlcExternalSourceSystemGroup

Example

```
// generate source for V14
var blocks = new List<PlcBlock>() {block1};
var types = new List<PlcBlock>() {udt1};
var fileInfoBlock = new FileInfo("D:\Export\Block.scl");
var fileInfoType = new FileInfo("D:\Export\Type.udt");

PlcBlocksSystemGroup blocksGroup = ...;
blocksGroup.GenerateSourceFromBlocks(blocks, fileInfo);
PlcTypesSystemGroup plcDataTypesGroup = ...;
plcDataTypesGroup.GenerateSourceFromTypes(types, fileInfo);

//generate source as of V14 SP1
var blocks = new List<PlcBlock>() {block1};
var types = new List<PlcBlock>() {udt1};
var fileInfoBlock = new FileInfo("D:\Export\Blocks.scl");
var fileInfoType = new FileInfo("D:\Export\Type.udt");

PlcExternalSourceSystemGroup externalSourceGroup = plc.ExternalSourceGroup;
externalSourceGroup.GenerateSource(blocks, fileInfoBlock);
externalSourceGroup.GenerateSource(types, fileInfoType);
```

9.3.4.3 Schema extension

Schema extension for comments and start values

Comments and start values are stored in new element called "Subelement" which refers to the array element with "Path" attribute.

Subelement contains start value and comment for the referenced array element. Attribute "Path" at StartValue is removed in the new schema.

Schema definition of "Subelement":

```
<xs:element name="Subelement" type="Subelement_T"/>
<xs:complexType name="Subelement_T">
  <xs:sequence>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="StartValue"/>
      <xs:element ref="Comment"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="Path" type="IndexPath_TP"/>
</xs:complexType>
```

Extension of member type:

```
<xs:complexType name="Member_T">
  <xs:sequence>
    <xs:element ref="AttributeList" minOccurs="0" maxOccurs="1"/>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="Member"/>
      <xs:element ref="Sections"/>
      <xs:element ref="StartValue"/>
      <xs:element ref="Comment"/>
      <xs:element ref="Subelement"/>
    </xs:choice>
  </xs:sequence>
</xs:complexType>
```

Examples:

Storage of comments and start values in simple arrays:

```
<Member Name="Static_1" Datatype="Array[0..1] of Bool">
  <Comment>
    <MultiLanguageText Lang="de-DE">comment for array</MultiLanguageText>
  </Comment>
  <Subelement Path="0">
    <StartValue>true</StartValue>
    <Comment>
      <MultiLanguageText Lang="de-DE">comment for array element 0</MultiLanguageText>
    </Comment>
  </Subelement>
  <Subelement Path="1">
    <StartValue>true</StartValue>
    <Comment>
      <MultiLanguageText Lang="de-DE">comment for array element 1</MultiLanguageText>
    </Comment>
  </Subelement>
</Member>
```

Storage of comments and start values in arrays of UDT:

```

<Member Name="Static_1" Datatype="Array[0..1] of &quot;User_data_type_1&quot;">
    <Comment>
        <MultiLanguageText Lang="de-DE">comment for array</MultiLanguageText>
    </Comment>
    <Subelement Path="0">
        <Comment>
            <MultiLanguageText Lang="de-DE">cmt array 0</MultiLanguageText>
        </Comment>
    </Subelement>
    <Sections>
        <Section Name="None">
            <Member Name="Element_1" Datatype="Bool">
                <Subelement Path="0">
                    <StartValue>true</StartValue>
                    <Comment>
                        <MultiLanguageText Lang="de-DE">comment for element 0</MultiLanguageText>
                    </Comment>
                </Subelement>
                <Subelement Path="1">
                    <StartValue>true</StartValue>
                    <Comment>
                        <MultiLanguageText Lang="de-DE">comment for element 1</MultiLanguageText>
                    </Comment>
                </Subelement>
            </Member>
            <Member Name="Element_2" Datatype="Struct">
                <Member Name="Element_1" Datatype="Int">
                    <Subelement Path="0">
                        <StartValue>11</StartValue>
                        <Comment>
                            <MultiLanguageText Lang="de-DE">comment for element 0</MultiLanguageText>
                        </Comment>
                    </Subelement>
                </Member>
                <Member Name="Element_2" Datatype="Bool">
                    <Subelement Path="1">
                        <StartValue>true</StartValue>
                        <Comment>
                            <MultiLanguageText Lang="de-DE">comment for element 1</MultiLanguageText>
                        </Comment>
                    </Subelement>
                </Member>
            </Section>
        </Sections>
    </Member>

```

Storage of comments and start values in arrays of struct:

```

<Member Name="Static_1" Datatype="Array[0..1] of Struct">
    <Member Name="Static_1" Datatype="Int">
        <Subelement Path="0">
            <StartValue>11</StartValue>
            <Comment>
                <MultiLanguageText Lang="de-DE">comment for int elem</MultiLanguageText>
            </Comment>
        </Subelement>
    </Member>
    <Member Name="Static_2" Datatype="Bool">
        <Subelement Path="1">
            <StartValue>true</StartValue>
            <Comment>
                <MultiLanguageText Lang="de-DE">comment for bool elem</MultiLanguageText>
            </Comment>
        </Subelement>
    </Member>
</Member>

```

9.3.4.4 Schema changes

Access node in SW.PlcBlocks.Access.xsd

Type attribute of the Access node has been moved to the children nodes of Access at

- AbsoluteOffset as required
- Address as optional

```
<StlStatement UID="22">
  <StlToken Text="L" />
  <Access Scope="Address">
    <Address Area="Local" Type="Word" BitOffset="80" />
  </Access>
</StlStatement>
```

Type attribute of Constant has been replaced with new ConstantType subnode.

```
<Access Scope="LocalConstant">
  <IntegerAttribute Name="NumBLs" Informative="true">5</IntegerAttribute>
  <Constant Name="LocalConstant_A">
    <ConstantType Informative="true">Int</ConstantType>
    <ConstantValue Informative="true">10</ConstantValue>
    <StringAttribute Name="Format" Informative="true">Dec_signed</StringAttribute>
  </Constant>
</Access>
```

The value of the Scope attribute in Access has been renamed to TypedConstant if the ConstantValue contains type qualified value(e.g.: int#10).

Constant does not have Type attribute if ConstantValue contains type qualified value (e.g.: int#10).

Local variables do not have Address node if Scope is LocalVariable.

If an Access is nested within another Access at any level, only the outer Access must have an UId.

Address node in SW.PlcBlocks.Access.xsd

BitOffset attribute of Address node became optional.

Declarations for exporting absolute access have changed as shown in the following table:

Area as of V14 SP1	Type	Block number	Bit offset	Example
DB	Block_DB	mandatory	forbidden	OPN %DB12
DB	unordered	existing	mandatory	%DB100.DBX10.3
DB	unordered	not existing	mandatory	%DB100.DBX10.3
L	unordered	forbidden	mandatory	%LW10.0
I	unordered	forbidden	mandatory	%I0.0
Q				%Q0.0
M				%M0.0

Area as of V14 SP1	Type	Block number	Bit offset	Example
T C	unordered	forbidden	mandatory	%T0 %C1
Block_FC Block_FB	Block_FC Block_FB	mandatory	forbidden	Call %FB4, %DB5 Input_1 := %FC10 Call %FB4, %DB5 Input_2 := %FB11
PeripheryInput	unordered	forbidden	mandatory	
Periphery Output	unordered	forbidden	mandatory	

Area node in SW.PlcBlocks.Access.xsd

Area node has got a simplified enum list:

- LocalC and LocalN became Local
- DBc, DBv, DBr are eliminated.

CallInfo node in SW.PlcBlocks.Access.xsd

Name attribute of CallInfo node became optional

BlockType attribute of CallInfo node became required

+2.2.5 User Block Calls

Constant node in SW.PlcBlocks.Access.xsd

Constant node references CostantType node with minOccurs=0

Constant node doesn't reference IntegerAttribute node any more

ConstantValue node in SW.PlcBlocks.Access.xsd

ConstantValue node gets an Informative attribute

Instruction node in SW.PlcBlocks.Access.xsd

Instruction node references Acces node with minOccurs=0

Parameter attributes Section, Type and TemplateReference have been deleted at Instruction.

Parameter node in SW.PlcBlocks.Access.xsd

SectionName attribute of the Parameter node became optional.

Values for Scope in SW.PlcBlocks.Access.xsd

Enum list of Scope has been extended with:

- TypedConstant
- AddressConstant
- LiteralConstant
- AlarmConstant
- Address
- Statusword
- Expression
- Call
- CallWithType

Statusword node in SW.PlcBlocks.Access.xsd

Enum list of Statusword has been extended with:

- STW

ConstantType node in SW.PlcBlocks.Access.xsd

New node ConstantType is introduced with optionally used attribute Informative.

CallRef node in SW.PlcBlocks.LADFBDB.xsd

CallRef node is renamed to Call and omits the BooleanAttribute subnode.

InstructionRef node in SW.PlcBlocks.LADFBDB.xsd

InstructionRef node is replaced by Part node

Part node in SW.PlcBlocks.LADFBDB.xsd

New node ConstantType is introduced and replaces the InstructionRef node

- Attributes: Name and Version
- Subnodes: Instruction subnode as new choice to existing Equation
- doesn't have neither BooleanAttribute subnode nor Gate attribute

Wire node in SW.PlcBlocks.LADFBDB.xsd

Name attribute of Wire node removed.

TemplateReference node in SW.PlcBlocks.LADFBDB.xsd

TemplateReference node is deleted.

StatementList node in SW.PlcBlocks.STL.xsd

Enum list of StatementList (STL_TE):

- L_STW has been removed
- T_STW has been removed

9.3.4.5 Behaviour changes

Absolute Access

In V14 the import of absolute access has been aborted for most combinations. As of V14 SP1 the import of absolute access works for the following areas:

- Input
- Output
- Memory
- Timer, if supported on the PLC
- Counter, if supported on the PLC
- DB
- DI

If a symbol access and an absolute access is used at the same time and is not rejected by schema or node kind validation the import will only succeed when both access informations are successfully resolved. When the symbol access leads to different information in comparison to the absolute access the import is rejected.

```

<Access Scope="Address">
    <Address Area="Memory" Type="Word" BitOffset="0" />
</Access>

<CallInfo Name="Block_1" BlockType="FC">
    <Parameter Name="Input_1" Section="Input" Type="Int" />
    <Parameter Name="Input_1" Section="Input" Type="Int" />
    <!-- Import will be aborted because parameter name 'Input_1' is used more than once -->
    <Parameter Name="Output_1" Section="NotExisting" Type="Int" />
    <!-- Import will be aborted because the section 'NotExisting' can not be used. -->
    <Parameter Name="ENO" Section="Output" Type="Int" />
    <!-- Import will be aborted because the parameter name 'ENO' is restricted and can not be used. -->
</CallInfo>

```

Indirect DB Access

As of V14 SP1 indirect DB Access can only be imported, when the 'offset', 'type' and 'symbol' are provided.

```
...
<Access Scope="LocalVariable" UID="21">
  <Symbol>
    <Component Name="Output_3" />
    <AbsoluteOffset BitOffset="16" Type="Word" />
  </Symbol>
</Access>
...
...
```

Symbolic and absolute information for local access

When importing "symbolic access" all possible provided "absolute access informations" are validated, if they are not flagged as "informative". As of V14 SP1 the import will be aborted when the absolute information does not match.

Block interface constraints

In V14SP1 several constraints are checked. These constraints are well known to users of the block interface editor. Whenever the block interface editor renames a parameter by adding or increasing '_1' the OPNS import will be aborted.

The following constraints are validated for instance:

- Duplicated parameter names
- Wrong section names. Including 'Return-Section' for FB blocks
- Restricted words

Sorting sections on import

When the called block does not exist at the time of import the interface definition at the call side will be used to display the called user block. In V14 SP1 the sections will be sorted in the order they would be displayed in the block interface of the called block, if it would have been existing with the same parameters.

The section order of the parameters imported is:

- Input
- Output

The following STL xml example

```

<StlStatement UIId="21">
<StlToken Text="CALL" />
<Access Scope="Call1">
  <CallInfo Name="Block_2" BlockType="FC">
    <Parameter Name="Output_1" Section="Output" Type="Int">
      <Access Scope="GlobalVariable">
        <Symbol>
          <Component Name="Tag_3" />
        </Symbol>
      </Access>
    </Parameter>
    <Parameter Name="Input_1" Section="Input" Type="Int">
      <Access Scope="GlobalVariable">
        <Symbol>
          <Component Name="Tag_1" />
        </Symbol>
      </Access>
    </Parameter>
    <Parameter Name="Output_2" Section="Output" Type="Int">
      <Access Scope="GlobalVariable">
        <Symbol>
          <Component Name="Tag_4" />
        </Symbol>
      </Access>
    </Parameter>
    <Parameter Name="Input_2" Section="Input" Type="Int">
      <Access Scope="GlobalVariable">
        <Symbol>
          <Component Name="Tag_2" />
        </Symbol>
      </Access>
    </Parameter>
  </CallInfo>
</Access>
</StlStatement>

```

will result in

CALL "Block_2" Input_1 := "Tag_1" Input_2 := "Tag_2" Output_1 := "Tag_3" Output_2 := "Tag_4"	
--	--

Unique user block callee names

In TIA Portal names have to be unique. This means for instance a tag can not have the same name like a block. For TIA Portal Openness API XML import this means when the XML contains a user block call, where the called block does not exist at the time of import, the name of the called block has to be unique to all existing names in the project. When the called block name is not unique the import will be aborted.

In the following example the import will be aborted, because the Name of the called Block "Tag_1" is already used for a tag table.

Major Changes

9.3 Major changes in V14 SP1

```
...
<SW.Tags.PlcTag ID="1" CompositionName="Tags">
  <AttributeList>
    <DataTypeName>Int</DataTypeName>
    <LogicalAddress>%MW2</LogicalAddress>
    <Name>Tag_1</Name>
  </AttributeList>
</SW.Tags.PlcTag>
...
...
<StlStatement UID="21">
  <StlToken Text="CALL" />
  <Access Scope="Call">
    <CallInfo Name="Tag_1" BlockType="FC">
      <Parameter Name="Input_1" Section="Input" Type="Int">
        <Access Scope="GlobalVariable">
          <Symbol>
            <Component Name="Tag_1" />
          </Symbol>
        </Access>
      </Parameter>
    ...
  
```

In the following example the import will be aborted, because two parameters have the same name ""Input1".

```
<StlStatement UID="22">
  <StlToken Text="CALL" />
  <Access Scope="Call">
    <CallInfo Name="Block_1" BlockType="FB">
      <Instance Scope="GlobalVariable">
        <Component Name="Block_1_DB" />
      </Instance>
      <Parameter Name="Input1" Section="Input" Type="Int">
        <Access Scope="GlobalVariable">
          <Symbol>
            <Component Name="Tag_9" />
          </Symbol>
        </Access>
      </Parameter>
      <Parameter Name="Input1" Section="Input" Type="Time">
        <Access Scope="TypedConstant">
          <Constant>
            <ConstantValue>T#1s</ConstantValue>
          </Constant>
        </Access>
      </Parameter>
    </CallInfo>
  </Access>
</StlStatement>
```

Library block calls

The imported XML may contain calls of user blocks. These user blocks are identified by name.

User blocks can also call library elements. These library elements can be generated as 'library block calls'. Because library blocks are using the same namespace as user blocks, the import of a user block call done by name can call the implementation of a library block.

Before V14 SP1 the import tried to map the parameters between the user block call and the instruction block call. Sometimes the import aborted, sometimes the import deleted all not matching parameters.

As of V14 SP1 the user block call will still find the library block, but the call will not become valid.

Block type mismatch

When the XML contains a user block call of 'Block_1' with more parameters as the corresponding FC in the project as of V14 SP1 the import defines a new called block interface matching the user block call from the XML. The next program block compile will attempt the call update.

New scopes for constants

With V14SP1 several new scopes for constants have been invented. The import only succeeds when the values in the xml match the constant scope. The import may abort when not all the provided information for a constant match the existing constant.

```
...
<Access Scope="LiteralConstant">
    <Constant>
        <ConstantType>Int</ConstantType>
        <ConstantValue>16#0000_0001</ConstantValue>
    </Constant>
</Access>
...
<Access Scope="TypedConstant">
    <Constant>
        <ConstantValue>Int#10</ConstantValue>
    </Constant>
</Access>
...
<Access Scope="LiteralConstant">
    <Constant>
        <ConstantType>Int</ConstantType>
        <ConstantValue>10</ConstantValue>
    </Constant>
</Access>
...
<Access Scope="GlobalConstant">
    <Constant Name="Constant_1" />
</Access>
...
<Access Scope="LocalConstant">
    <Constant Name="Constant_1" />
</Access>
...
<Access Scope="AddressConstant">
    <Constant Name="Tag_1" />
</Access>
...
<Access Scope="AlarmConstant">
    <Constant>
        <ConstantType>C_Alarm</ConstantType>
        <ConstantValue>16#0000_0001</ConstantValue>
    </Constant>
</Access>
...
```

Instruction version annotation

As of V14 SP1 only instruction versions usable on the PLC to import to can be imported. When no instruction version is annotated in the xml the version selected in the PLC will be used. In LAD and FBD some elements represented as instruction do not use versioning. These elements can only be imported without version.

```
...
<Part Name="MIN" Version="1.0" UIId="27" DisabledENO="false">
  <TemplateValue Name="card" Type="Cardinality">2</TemplateValue>
  <TemplateValue Name="value_type" Type="Type">Int</TemplateValue>
</Part>
<Part Name="MIN" UIId="28" DisabledENO="false">
  <TemplateValue Name="card" Type="Cardinality">2</TemplateValue>
  <TemplateValue Name="value_type" Type="Type">Int</TemplateValue>
</Part>
...
```

Disabled ENO

The "disabled ENO" feature is used on 1200 and 1500 PLC to deactivate runtime consuming ENO connection state calculation.

As of V14 SP1 the DisabledENO flag can only be imported on PLC's supporting the feature.

```
...
<Part Name="Add" UIId="24" DisabledENO="false">
  <TemplateValue Name="Card" Type="Cardinality">2</TemplateValue>
  <TemplateValue Name="SrcType" Type="Type">Int</TemplateValue>
</Part>
...
```

Type validation for absolute L-Stack access

As of V14 SP1 the import is aborted when the type can not be used or mapped.

Validation of index idents

Index access's are usable where 'symbolic access to memory' is defined. For instance, local access, global access, indirect access.

When a literal constant is used as index, the singed and unsigned integer types are changed to Dint. As of V14 SP1 the import is aborted when a type outside the mentioned range is provided.

All index access's are checked, whether the kind of access can be used as 'index access' at all. As of V14 SP1 the import is aborted when the defined index access can not be used.

Element order sorting

As of V14 SP1 the elements in LAD and FBD will be sorted 'code generation order' where automatically possible during export. In some very rare cases the exported XML is not importable again. In these cases either the XML has to be adapted or the corresponding networks have to be deleted and reprogrammed. But the order of wires and references is still not reliable.

Alarm Constants

With V14 SP1 the compile checks for valid alarm constants have been extended. It might occur that projects are not compilable in V14 SP1 due to an xml imported in V14 with broken alarm constants. In this case open the relevant network in LAD/FDB editor and delete the alarm actual operand. The editor will automatically recreate a valid alarm constant.

```
<FlgNet>
  <Parts>
    <Access Scope="AlarmConstant" UIId="21">
      <Constant>
        <ConstantType>C_Alarm</ConstantType>
        <ConstantValue>16#0000_0002</ConstantValue>
      </Constant>
    </Access>
    <Call UIId="22">
      <CallInfo Name="Block_1" BlockType="FB">
        <Instance UIId="23" Scope="GlobalVariable">
          <Component Name="Block_1_DB" />
        </Instance>
        <Parameter Name="Input_1" Section="Input" Type="C_Alarm" />
      </CallInfo>
    </Call>
  </Parts>
  <Wires>
    <Wire UIId="24">
      <PowerRail />
      <NameCon UIId="22" Name="en" />
    </Wire>
    <Wire UIId="25">
      <IdentCon UIId="21" />
      <NameCon UIId="22" Name="Input_1" />
    </Wire>
  </Wires>
</FlgNet>
```

Constraints for instances of user blocks and instructions

In V14 it was possible to import user FC block calls with an instance and sometimes even compile these calls.

As of V14 SP1 the import of instances is only possible where instances are supported. Existing projects with instances at FC user block calls and instructions may not compile any more. In this case the call has to be deleted and reprogrammed. Any attempt to do a call update or 'other means of automated repair' will fail.

EnEno visible

In V14 the EN and ENO connections of 'InstructionRef' have been usable or not depending on the ENENO flag.

As of V14SP1 the OPNS during the import based on the element and the wiring either the EN and ENO connections are used. Because of this automatic detection a different EN and ENO connection usage can be noticed. Most probably only the IEC timer and IEC counter boxes may show some issues.

UId assignment

The assignment of UIds to parts, access and wires changes with V14 SP1. The UIds for statements, CallInfo and operands have to be unique within a compile unit. From TIA Portal point of view the UIds in the XML are keys, without any additional meaning besides identifying an element.

Checking of character strings

More strict checks concerning quotation marks, surrogate characters and control characters are performed for the Name attribute during the import of

- IntegerAttribute
- StringAttribute
- DateAttribute
- AutomaticTyped
- Component
- Invisible
- Label
- NameCon
- Negated
- TemplateValue
- CallInfo
- Instruction
- Parameter
- Part
- Step

More strict checks concerning surrogate characters and control characters are performed during the import of

- Titles of blocks and networks
- LineComment text
- Constant strings (String, WString, Char, Wchar typed)

More strict checks concerning surrogate characters and control characters (tab and new line allowed) are performed during the import of

- Comments of blocks and networks
- String attributes
- Nodes defining multilanguage texts, e.g. Alarmtext, Comments
- Token texts

Case insensitivity of template operations and parameters

As of V14 SP1 case insensitivity of template operations for instructions and call or instruction parameters will be imported and automatically corrected.

The following code will be imported and the incorrect value "Eq" will be corrected to "EQ" and the incorrect parameter "iN1" will be corrected to "IN1":

```
<StlStatement UIId="22">
  <StlToken Text="CALL" />
  <Access Scope="Call">
    <Instruction Name="CompType">
      <TemplateValue Name="src_type" Type="Type">Variant</TemplateValue>
      <TemplateValue Name="relation" Type="Operation">Eq</TemplateValue>
      <Parameter Name="iN1">
        <Access Scope="GlobalVariable">
          <Symbol>
            <Component Name="Tag_12" />
          </Symbol>
        </Access>
      </Parameter>
      ...
    </Instruction>
  </Access>
</StlStatement>
```

Multiinstances used in calls

As of V14 SP1 the import is aborted if the multiinstance used in a call does not exists.

The following code shows an xml example where the multiinstance is defined correctly in the interface section:

```

<SW.Blocks.FB ID="0">
  <AttributeList>
    <Interface>
      <Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
        <Section Name="Input" />
        <Section Name="Output" />
        <Section Name="InOut" />
        <Section Name="Static">
          <!-- The next line must be present if multiinstance is used in code-->
          <Member Name="Static_1" Datatype=""Block_2"" />
        </Section>
      ....
    <StlStatement UID="22">
      <StlToken Text="CALL" />
      <Access Scope="Call">
        <CallInfo BlockType="FB">
          <!-- Multiinstance usage-->
          <Instance Scope="LocalVariable">
            <Component Name="Static_1" />
          </Instance>
          <Parameter Name="Input_1" Section="Input" Type="Int">
            <Access Scope="GlobalVariable">
              <Symbol>
                <Component Name="Tag_9" />
              </Symbol>
            </Access>
          </Parameter>
        </CallInfo>
      </Access>
    </StlStatement>
  </AttributeList>
</SW.Blocks.FB>

```

Template cardinalities in STL

In STL the template cardinalities for every instruction has a fixed default value which is the only valid value. As of V14 SP1 the import is aborted if another value is used for the cardinality.

Importing indirect access

As of V14 SP1 indirect access can only be imported where they can be compiled.

```

<StlStatement UID="22">
  <StlToken Text="L" />
  <Access Scope="Address">
    <Indirect Width="Word" Area="Memory">
      <Access Scope="LocalVariable">
        <Symbol>
          <Component Name="Temp_1" />
        </Symbol>
      </Access>
    </Indirect>
  </Access>
</StlStatement>

```

Importing statuswords

As of V14 SP1 the statusword can only be imported at statements where they are supported.

- L - Supported statusword: STW
- T - Supported statusword: STW
- A - Supported statusword: BR, OV, OS, EQ, NE, GT, Lt, GE, LE, U0, NU
- AN - Supported statusword: BR, OV, OS, EQ, NE, GT, Lt, GE, LE, U0, NU
- O - Supported statusword: BR, OV, OS, EQ, NE, GT, Lt, GE, LE, U0, NU
- ON - Supported statusword: BR, OV, OS, EQ, NE, GT, Lt, GE, LE, U0, NU
- X - Supported statusword: BR, OV, OS, EQ, NE, GT, Lt, GE, LE, U0, NU
- XN - Supported statusword: BR, OV, OS, EQ, NE, GT, Lt, GE, LE, U0, NU

Note

Most statuswords are only useful on 300 and 400 plcs.

Empty statements

The import is aborted if a statement does not have a node <StlStatement/>. In case of an empty statement, add the <StlToken Text="Empty_Line" /> node.

The import is aborted if an empty statement has comments. For a statement with only comments use the <StlToken Text="COMMENT" />.

```
<!-- Declaration of an empty statement -->
<StlStatement UIId="23">
  <StlToken Text="EMPTY_LINE" />
</StlStatement>

<!-- Declaration of a statement with only comments-->
<StlStatement UIId="22">
  <LineComment>
    <Text>Comment number 1</Text>
  </LineComment>
  <StlToken Text="COMMENT" />
</StlStatement>
```

9.3.4.6 Block attribute changes

Changes in general attributes

AutoNumber has got a new default value (false) at classic OBs

HeaderVersion has got a new type System.Version (instead of String)

IsKnowHowProtected is applied for user defined data types as well

ILibraryTypeInstance.ConnectedVersion, ILibraryTypeInstance.Dependencies, ILibraryTypeInstance.Dependents are eliminated from the table of general attributes because they are neither exported in XML nor accessible via API.

MemoryLayout gets new default: Standard in classic PLCs and Optimized on plus PLCs

Number is applied for user defined data types and it is represented in XML and accessible via API as well

Changes in specific attributes

IsOnlyStoredInLoadMemory and IsWriteProtectedInAS became read-only for IDBofUDT if it belongs to a system library element.

OfSystemLibElement and OfSystemLibVersion are relocated from general to specific attributes

OfSystemLibVersion has got a new type System.Version (instead of String)

ParameterPassing remains read-write at FCs and FBs only if

- ProgrammingLanguage is STL and
- MemoryLayout is standard and
- interface is empty

GraphVersion has got a new type System.Version (instead of String)

a new attribute called ExtensionBlockName is introduced for FBs written in Graph as of Graph version V4

a new attribute called InvalidValuesAcquisition is introduced for FBs written in Graph as of Graph version V4

a new attribute called IsWriteProtected is introduced for code blocks

DownloadWithoutReinit became read-only and also applied for IDBofFBs

Supervisions became read-only on IDBofFBs .

Changes in enums

The enum values for ProgrammingLanguage are changed as follows:

- a new enum value F_CALL is introduced
- a new enum value Motion_DB is introduced for Motion technological object
- GRAPH_SEQUENCE, GRAPH_ACTIONS, GRAPH_ADDINFOS are deleted from the enum. They are replaced with GRAPH.

The enum values for BlockType are changed as follows:

- the values OB, FC, DB, SFC are deleted because this enum is only used at InstanceOfType attribute

9.4 Major changes in V14

9.4.1 Major changes of the object model

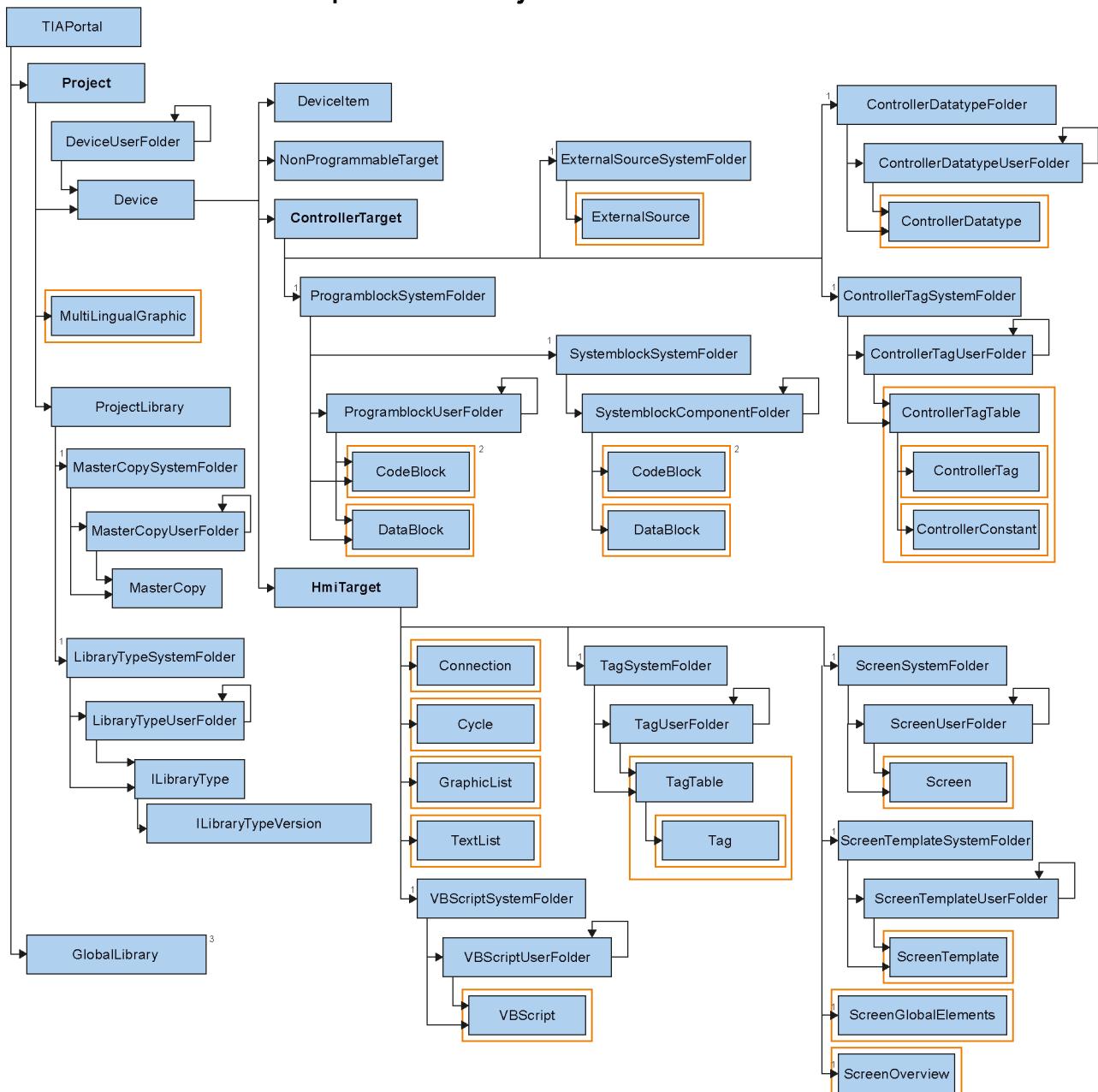
Object model of TIA Portal Openness V13 SP1 and older

In order to allow you a comparison between the old and the new object model of TIA Portal Openness, the diagram below describes the object model of TIA Portal V13 SP1.

Note

The object model described on the diagram is obsolete, for information about the object model of TIA Portal Openness V14 SP1 refer to AUTOHOTSPOT

Openness object model V13 SP1



9.4.2 Before updating an application to TIA Portal Openness V14

Application

Before updating an application to TIA Portal Openness V14 change the following settings:

1. Adapt the references to the V14 API by adding the following TIA Portal Openness APIs:
 - Siemens.Engineering
 - Siemens.Engineering.Hmi
2. Change the .Net framework of your Visual Studio to version 4.6.1
3. Update the assembly resolve method by adapting the new installation path of the TIA Portal.
 - If you have evaluated from the registry, adapt the new key, according to the following example:
"HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\Automation_InstalledSW\\TIAP14\TIA_Opns\..."
 - If you are using the application configuration file, adapt the paths to the new installation path.

9.4.3 Major string changes

Introduction

The following changes were made in TIA Portal Openness V14, which may impact your existing applications:

Change	Required program code adjustment
Compile methods have been changed.	<p>Change the compile methods according to the following example:</p> <ul style="list-style-type: none">• TIA Portal Openness V13 SP1(obsolete): controllerTarget.Compile(CompilerOptions.Software, BuildOptions.Rebuild);• TIA Portal Openness V14: plcSoftware.GetService<ICompilable>().Compile();
New namespaces have been added.	<ol style="list-style-type: none">1. Add the following namespace statements: Siemens.Engineering.SW.Blocks; Siemens.Engineering.SW.ExternalSources; Siemens.Engineering.SW.Tags; Siemens.Engineering.SW.Types;2. Remove the using ControllerTarget = Siemens.Engineering.HW.ControllerTarget namespace statement.3. Compile the application.

Change	Required program code adjustment
ControllerTarget has been replaced by PlcSoftware and functionality has been changed in some cases.	<ol style="list-style-type: none"> 1. Review the code examples in the documentation which belong to your application functionality. 2. Update the program code of your TIA Portal Openness application according to the following example: <ul style="list-style-type: none"> - TIA Portal Openness V13 SP1(obsolete): ControllerTarget controllerTarget = deviceItem as ControllerTarget - TIA Portal Openness V14: PlcSoftware plcSoftware = deviceItem.GetService<SoftwareContainer>().Software as PlcSoftware 3. Compile the application.
Objects have been replaced.	<ol style="list-style-type: none"> 1. Search and replace the following objects: <pre>DeviceUserFolderAggregation = DeviceUserGroupComposition DeviceFolders = DeviceGroups DeviceUserFolder = DeviceUserGroup ProgramblockSystemFolder = PlcBlockSystemGroup ProgramblockUserFolder = PlcBlockUserGroup IBlock = PlcBlock ControllerDatatypeSystemFolder = PlcTypeSystemGroup ControllerDatatypeUserFolder = PlcTypeUserGroup ControllerDatatype = PlcType ControllerTagSystemFolder = PlcTagTableSystemGroup ControllerTagUserFolder = PlcTagTableUserGroup ControllerTagTable = PlcTagTable ControllerTag = PlcTag ControllerConstant = PlcConstant ExternalSourceSystemFolder = PlcExternalSourceSystemGroup ExternalSource = PlcExternalSource IONline = OnlineProvider ILibraryType = LibraryType</pre> 2. Compile the application.

Major Changes

9.4 Major changes in V14

Change	Required program code adjustment
Aggregations have been replaced by compositions.	<ol style="list-style-type: none">1. Replace every Aggregation of your code by Composition according to the following examples: ProjectAggregation = ProjectComposition IDeviceAggregation = IDeviceComposition TagTableAggregation = TagTableComposition CycleAggregation = CycleComposition GraphicListAggregation = GraphicListComposition TextListAggregation = TextListComposition ConnectionAggregation = ConnectionComposition MultiLingualGraphicAggregation = MultiLingualGraphicComposition UpdateCheckResultMessageAggregation = UpdateCheckResultMessageComposition2. Compile the application.
Folders have been replaced by groups in every relationship except HMI devices.	<ol style="list-style-type: none">1. Replace every Folder in your program code by Group except code parts which concern to HMI devices.2. Compile the application.
The GetAttributeNames method has been replaced by the GetAttributeInfos method.	<ol style="list-style-type: none">1. Use <code>IList<EngineeringAttributeInfo> IEngineeringObject.GetAttributeInfos(AttributeAccessMode attributeAccessMode);</code> to determine attributes.2. Compile the application. For more detailed information, refer to Determining the object structure and attributes (Page 121).
The Close method for closing an object has changed.	<ol style="list-style-type: none">1. Replace <code>project.Close(CloseMode.PromptIfModified);</code> by <code>project.Close();</code>.2. Compile the application. For more detailed information, refer to Closing a project (Page 133).

Change	Required program code adjustment
Simultaneous access has been replaced by exclusive access and transactions.	<p>1. Replace simultaneous access by exclusive access and transactions according to the following examples:</p> <ul style="list-style-type: none"> - TIA Portal Openness V13 SP1(obsolete): <pre>tiaProject.StartTransaction("Resetting project to default"); ... tiaProject.CommitTransaction();</pre> - TIA Portal Openness V14: <pre>//Use exclusive access to avoid user changes ExclusiveAccess exclusiveAccess = tiaPortal.ExclusiveAccess(); ... exclusiveAccess.Dispose(); //Use transaction to be able to rollbank changes: Transaction transaction = exclusiveAccess.Transaction(tiaProject, "Compiling device"); transaction.CommitOnDispose();</pre> <p>2. Compile the application. See Exclusive access (Page 91) and Transaction handling (Page 100) for further information.</p>
Online access to the CPU has been changed	<p>1. Change the online access to the CPU according to the following examples:</p> <ul style="list-style-type: none"> - TIA Portal Openness V13 SP1(obsolete): <pre>((IOnline)controllerTarget).GoOffline() ;</pre> - TIA Portal Openness V14: <pre>((DeviceItem) plcSoftware.Parent.Parent).GetService<O nlin eProvider>().GoOffline();</pre> <p>2. Compile the application.</p>
The hardware configuration has been changed	<p>1. Change the hardware configuration: <code>Device.Elements = Device.Items</code></p> <p>2. Remove the following hardware attributes:</p> <ul style="list-style-type: none"> - Device.InternalDeviceItem - Device.SubType <p>3. Compile the application.</p>

See also

[Handling exceptions \(Page 740\)](#)

[Connecting to the TIA Portal \(Page 76\)](#)

9.4.4 Import of files generated with TIA Portal Openness V13 SP1 and previous**Application**

When you try to import files which were generated with TIA Portal Openness V13 SP1 or previous an exception will be thrown because of incompatibility. This is caused by changes on HMI tags and HMI screen items. The following tables are showing the main attribute changes, for more detailed information refer to the chapter "Creating screens Working with objects and object groups > Working with objects > Configuring ranges" of the TIA Portal online help:

Changes of HMI tags

The following table shows the main changes of HMI tag attributes:

Removed attributes	Added attributes
RangeMaximumType	LimitUpper2Type.
RangeMaximum	LimitUpper2.
RangeMinimumType	LimitLower2Type.
RangeMinimum	LimitLower2.
	LimitUpper1Type
	LimitUpper1
	LimitLower1Type
	LimitLower1

Changes of HMI screen items

The following table shows the main changes of slider attributes:

Removed attributes	Added attributes
	RangeLower1Color RangeLower1Enabled RangeLower2Color RangeLower2Enabled RangeNormalColor RangeNormalEnabled RangeUpper1Color RangeUpper1Enabled RangeUpper2Color RangeUpper2Enabled ScalePosition ShowLimitLines ShowLimitMarkers ShowLimitRanges

The following table shows the main changes of gauge attributes:

Removed attributes	Added attributes
DangerRangeColor	RangeLower1Color
DangerRangeStart	RangeLower1Enabled
DangerRangeVisible	RangeLower2Color
WarningRangeColor	RangeLower2Enabled
WarningRangeStart	RangeNormalColor
WarningRangeVisible	RangeNormalEnabled
	RangeUpper1Color
	RangeUpper1Enabled
	RangeUpper1Start
	RangeUpper2Color
	RangeUpper2Enabled
	RangeUpper2Start

The following table shows the main changes of bar attributes:

Removed attributes	Added attributes
AlarmLowerLimitColor	RangeLower1Color
AlarmUpperLimitColor	RangeLower1Enabled
	RangeLower2Color
	RangeLower2Enabled
	RangeNormalColor
	RangeNormalEnabled
	RangeUpper1Color
	RangeUpper1Enabled
	RangeUpper2Color
	RangeUpper2Enabled

Index

"

"Devices & networks" editor

Open, 178

"Tags" editor

Starting, 565

A

Accessing

Master copy in project library, 163

Acknowledging system events program-controlled, 84

B

Basic structure of an AML export file, 891

Basic structure of an export file, 763, 895

Block

Creating group, 507

Deleting, 507

Deleting group, 508

Exporting, 868

Generate source, 514

Importing, 867

Querying information, 503

Block editor

Starting, 521

C

Compiling

Hardware, 129

Software, 129

Technology object, 531

Technology object group, 532

Configuration

Your Openness application and the TIA Portal run on different computers, 47

Connecting

Analog drives by data block, 543

Analog drives by hardware address, 541

Cam track, 558

Drives, 551

Encoders, 556

Encoders by data block, 544

Encoders for analog drives by hardware address, 542

Encoders for PROFIdrives by hardware address, 540

Measuring input, 559

Output cam, 558

PROFIdrives by data block, 543

PROFIdrives by hardware address, 539

Synchronous axis with leading values, 560

telegram 750, 554

Connection to the TIA Portal

Close, 85

Setting up, 76

Copy

Content of a master copy in project folder, 166

Master copy, 169

Create

User-defined folders for HMI tags, 267

User-defined screen folders, 261

User-defined script folders, 269

Creating

Cam track, 545

Group for block, 507

Measuring input, 545

Output cam, 545

Technology object, 530

User-defined folder for PLC tag tables, 568

D

Data types

Technology object, 528

Deleting

All screens, 263

Block, 507

Connection, 266

Cycle, 264

Deleting a PLC tag table from a folder, 572

Graphic list, 266

Group for block, 508

Individual tag in a PLC tag table, 574

Individual tags of a tag table, 268

PLC constants, 575

Program block, 507

Project graphics, 129

Screen, 262

Screen template, 262

Tag table, 269

Technology object, 531

- Text list, 265
 - User data type, 520
 - User-defined folder for PLC tag tables, 569
 - VB script from a folder, 270
- E**
- Editing situation
 - Your Openness application and the TIA Portal run on the same computer, 48
 - Enumerating
 - All tags of a tag table, 267
 - Blocks, 501
 - Device items, 243
 - Devices, 228, 231
 - Multilingual texts, 119, 124
 - Parameter of technology object, 534
 - PLC tag tables, 569
 - PLC tags, 573
 - System subfolders, 499
 - Technology object, 533
 - User-defined block folders, 500
 - User-defined folder for PLC tags, 567
 - Enumerating device items, 243
 - Enumerating devices, 228, 231
 - Enumerating multilingual texts, 119, 124
 - Establishing a connection to the TIA Portal, 76
 - Example program, 51
 - Exceptions
 - When accessing the TIA Portal via public APIs, 740
 - Export file
 - Basic structure, 763, 891, 895
 - Contents, 752
 - Structure of the XML file, 763, 895
 - Export/import
 - Application, 38
 - Exportable screen objects, 785
 - Exporting
 - Block, 868
 - Individual tag or constant from a PLC tag table, 884
 - User data type, 868
- F**
- Finding
 - Cam track, 545
 - Measuring input, 545
 - Output cam, 545
 - Parameter of technology object, 535
 - Technology object, 533
 - Folders
 - Deleting, 175
 - Functions, 51
 - Closing a project, 133
 - Creating a user-defined folder for PLC tag tables, 568
 - Creating user-defined folders for HMI tags, 267
 - Creating user-defined screen folders, 261
 - Creating user-defined script subfolders, 269
 - Deleting a connection, 266
 - Deleting a cycle, 264
 - Deleting a graphic list, 266
 - Deleting a PLC tag table, 572
 - Deleting a screen, 262
 - Deleting a screen template, 262
 - Deleting a tag from a PLC tag table, 574
 - Deleting a tag from a tag table, 268
 - Deleting a tag table, 269
 - Deleting a text list, 265
 - Deleting a user-defined folder for PLC tag tables, 569
 - Deleting a VB script from a folder, 270
 - Deleting all screens, 263
 - Deleting project graphics, 129
 - Determining the system folder, 498
 - Enumerating blocks, 501
 - Enumerating device items, 243
 - Enumerating devices, 228, 231
 - Enumerating multilingual texts, 119, 124
 - Enumerating PLC tag tables in folders, 569
 - Enumerating PLC tags, 573
 - Enumerating system subfolders, 499
 - Enumerating tags of an HMI tag table, 267
 - Enumerating user-defined block folders, 500
 - Enumerating user-defined folders for PLC tags, 567
 - Exporting a tag or constant from a PLC tag table, 884
 - General, 76, 84, 85
 - General TIA portal settings, 111
 - HMI, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270
 - Importing a tag into a PLC tag table, 885
 - Importing PLC tag tables, 883
 - Limitation to projects of TIA Portal V13, 106
 - Open project, 106
 - PLC, 498, 499, 500, 501, 503, 568, 569, 572, 574, 575, 883, 884, 885
 - PLC constants, 575

Projects, 106, 111, 119, 124, 129, 131, 133, 178, 228, 231, 243, 566, 567, 569, 570, 573
 Public API application example, 69
 Querying information from a PLC tag table, 570
 Querying PLC and HMI targets, 178
 Querying system folders for PLC tags, 566
 Querying the "Program blocks" folder, 498
 Querying the block author, 503
 Querying the block family, 503
 Querying the block name, 503
 Querying the block number, 503
 Querying the block title, 503
 Querying the block type, 503
 Querying the block version, 503
 Querying the consistency attribute of a block, 503
 Querying the time stamp of a block, 503
 Reading the time of the last changes to a PLC tag table, 572
 Saving a project, 131

G

General TIA portal settings, 111
 Generate
 source from block, 514
 source from user data type, 514
 Global library
 Accessing, 137, 141
 Accessing language settings, 139

H

Hardware
 Compiling, 129
 Hierarchy of hardware objects of the object model, 66
 HMI tags of the "UDT" data type, 774

I

Import/Export
 Advanced XML formats for export/import of text lists, 780
 Also export default values, 752
 Basics, 747
 Data structure, 763, 895
 Editing an XML file, 751
 Export format, 749
 Export scope, 752
 Export settings, 752
 Exportable objects, 747

Exportable screen objects, 785
 Exporting a screen from a screen folder, 790
 Exporting a screen with a faceplate instance, 805
 Exporting a selected tag, 772
 Exporting a tag from a tag table, 772
 Exporting all graphics of a project, 756
 Exporting all screen templates, 796
 Exporting blocks with know-how protection, 826
 Exporting blocks without know-how protection, 868
 Exporting configuration data, 752
 Exporting connections, 783
 Exporting cycles, 766
 Exporting graphic lists, 782
 Exporting HMI tag tables, 768
 Exporting multilingual comments, 963, 972, 979, 980
 Exporting only modified values, 752
 Exporting permanent areas, 794
 Exporting PLC tag table, 882
 Exporting pop-up screens, 800
 Exporting screen templates, 797
 Exporting screens of an HMI device, 789
 Exporting slide-in screen, 803
 Exporting system blocks, 864
 Exporting tags, 982
 Exporting text lists, 778
 Exporting VB scripts, 775, 776
 Field of application, 750
 Graphics, 755
 HMI, 766, 767, 768, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 782, 784, 785, 789, 790, 792, 794, 795, 796, 797, 799, 800, 801, 803, 804, 805, 807, 882
 Importable objects, 747
 Importing a graphic list, 782
 Importing a screen including a faceplate instance, 807
 Importing an HMI tag into a tag table, 773
 Importing configuration data, 753
 Importing connections, 784
 Importing cycles, 767
 Importing graphics to a project, 757
 Importing multilingual comments, 963, 972, 979, 980
 Importing permanent areas, 795
 Importing pop-up screen, 801
 Importing screen templates, 799
 Importing screens to an HMI device, 792
 Importing slide-in screen, 804
 Importing tag table to a tag folder, 771
 Importing tags, 982

- Importing text list, 779
 - Importing VB scripts, 777
 - Objects of AML, 891
 - PLC, 826, 864, 868
 - Procedure for importing, 754
 - Project data, 756, 757
 - Restricting exports to modified values, 752
 - Restrictions, 749
 - Round trip devices and modules, 932
 - Setting the import behavior by means of program codes, 754
 - Special considerations for integrated HMI tags, 774
 - Stable AML GUIDs, 932
 - Importing
 - An individual tag into a PLC tag table, 885
 - Block, 867
 - PLC tag tables, 883
 - User data type, 887
 - Installation
 - Access authentication check, 31
 - Adding users to the user group, 31
 - Standard steps for accessing the TIA Portal, 36
 - TIA Openness V13 add-on package, 30
 - Installing the add-on package, 30
 - Instances
 - Determining type versions, 169
 - Integrated HMI tags, 774
-
- ## L
- Library
 - Accessing folders, 148
 - Determining type versions of instances, 169
 - Functions, 136
-
- ## M
- Master copies
 - Deleting, 175
 - Master copy
 - Copy content to project folder, 166
 - Copying, 169
-
- ## O
- Object model, 52
 - Objects
 - Exportable objects, 747
 - Importable objects, 747
-
- ## P
- Open
 - "Devices & networks" editor, 178
 - Opening a project, 106
-
- ## PLC
- Comparing, 478
 - Comparison with actual status, 478
 - Determining status, 425
 - Disconnecting an online connection, 482
 - Establishing an online connection, 482
- Program block
 - Deleting, 507
 - Programming overview, 51
- Project
 - Close, 133
 - Open, 106
 - Querying HMI targets, 178
 - Querying PLC targets, 178
 - Querying the device type, 178
 - Save, 131
 - Project library
 - Accessing, 137, 141
 - Accessing master copies, 163
 - Public API application example, 69
-
- ## Q
- Querying
 - Block author, 503
 - Block family, 503
 - Block name, 503
 - Block number, 503
 - Block title, 503
 - Block type, 503
 - Block version, 503
 - Consistency attribute of a block, 503
 - Finding the, 498
 - Information from a PLC tag table, 570
 - Information of block, 503

Information of user data type, 503
 Program blocks folder, 498
 System folders for PLC tags, 566
 Technology object, 529
 Time stamp of a block, 503

R**Read**

Time of the last changes to a tag table, 572

Reading

Parameter of technology object, 536

S

Saving a project, 131
 Siemens.Engineering, 43
 Siemens.Engineering.Hmi, 43
 Siemens.Engineering.Hmi.Communication, 43
 Siemens.Engineering.Hmi.Cycle, 43
 Siemens.Engineering.Hmi.Globalization, 43
 Siemens.Engineering.Hmi.RuntimeScripting, 43
 Siemens.Engineering.Hmi.Screen, 43
 Siemens.Engineering.Hmi.Tag, 43
 Siemens.Engineering.Hmi.TextGraphicList, 43
 Siemens.Engineering.HW, 43
 Siemens.Engineering.SW, 43
Software
 Compiling, 129
 Special considerations for HMI tags of the "UDT" data type, 774
Starting
 "Tags" editor, 565
 Block editor, 521
Status (PLC)
 Determining, 425
 Structure of the export data, 763, 891, 895

T

Technology object, 526
 Compiling, 531
 Creating, 530
 Data types, 528
 Deleting, 531
 Enumerating, 533
 Finding, 533
 Querying, 529
Technology object group
 Compiling, 532
 Terminating the connection to the TIA Portal, 85

TIA Portal Openness, 45
 Access, 37
 Access rights, 31
 Adding users to the user group, 31
 Basic concepts of aggregations, 96
 Basic concepts of associations, 96
 Basic concepts of object equality verification, 97
 Basic concepts when handling exceptions, 740
 Configuration, 47
 Export/import, 38
 Functional scope, 45
 Functions, 51
 Introduction, 45
 Necessary user knowledge, 29
 Programming overview, 51
 Public API, 51
 Requirements, 29
 Standard steps for accessing the TIA Portal, 36
 Typical tasks, 37
Types
 Deleting, 175

U

User data type
 Deleting, 520
 Exporting, 868
 Generate source, 514
 Importing, 887
 Querying information, 503

W

Writing
 Parameter of technology object, 537

X

XML file
 Edit, 751
 Export, 752

