SIEMENS EDA

Xpedition® Designer Automation Reference

Release X-ENTP VX.2.13
Document Revision 10

SIEMENS

Unpublished work. © 2023 Siemens

This Documentation contains trade secrets or otherwise confidential information owned by Siemens Industry Software Inc. or its affiliates (collectively, “Siemens”), or its licensors. Access to and use of this Documentation is strictly limited as set forth in Customer’s applicable agreement(s) with Siemens. This Documentation may not be copied, distributed, or otherwise disclosed by Customer without the express written permission of Siemens, and may not be used in any way not expressly authorized by Siemens.

This Documentation is for information and instruction purposes. Siemens reserves the right to make changes in specifications and other information contained in this Documentation without prior notice, and the reader should, in all cases, consult Siemens to determine whether any changes have been made.

No representation or other affirmation of fact contained in this Documentation shall be deemed to be a warranty or give rise to any liability of Siemens whatsoever.

If you have a signed license agreement with Siemens for the product with which this Documentation will be used, your use of this Documentation is subject to the scope of license and the software protection and security provisions of that agreement. If you do not have such a signed license agreement, your use is subject to the Siemens Universal Customer Agreement, which may be viewed at https://www.sw.siemens.com/en-US/sw-terms/base/uca/, as supplemented by the product specific terms which may be viewed at https://www.sw.siemens.com/en-US/sw-terms/supplements/.

SIEMENS MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS DOCUMENTATION INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY. SIEMENS SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL OR PUNITIVE DAMAGES, LOST DATA OR PROFITS, EVEN IF SUCH DAMAGES WERE FORESEEABLE, ARISING OUT OF OR RELATED TO THIS DOCUMENTATION OR THE INFORMATION CONTAINED IN IT, EVEN IF SIEMENS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

TRADEMARKS: The trademarks, logos, and service marks (collectively, "Marks") used herein are the property of Siemens or other parties. No one is permitted to use these Marks without the prior written consent of Siemens or the owner of the Marks, as applicable. The use herein of third party Marks is not an attempt to indicate Siemens as a source of a product, but is intended to indicate a product from, or associated with, a particular third party. A list of Siemens' Marks may be viewed at: www.plm.automation.siemens.com/global/en/legal/trademarks.html. The registered trademark Linux® is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide basis.

About Siemens Digital Industries Software

Siemens Digital Industries Software is a leading global provider of product life cycle management (PLM) software and services with 7 million licensed seats and 71,000 customers worldwide. Headquartered in Plano, Texas, Siemens Digital Industries Software works collaboratively with companies to deliver open solutions that help them turn more ideas into successful products. For more information on Siemens Digital Industries Software products and services, visit www.siemens.com/plm.

Support Center: support.sw.siemens.com
Send Feedback on Documentation: support.sw.siemens.com/doc_feedback_form

Revision History ISO-26262

Revision	Changes	Status/Date
10	Modifications to improve the readability and comprehension of the content. Approved by Kevin Chupp. All technical enhancements, changes, and fixes listed in the *Xpedition Enterprise Flow Release Notes* for this product are reflected in this document. Approved by Mike Bare.	Released March 2023
9	Modifications to title page to reflect the latest product version supported. Approved by Kevin Chupp. No technical enhancements, changes, and fixes listed in the *Xpedition Enterprise Flow Release Notes* for this product are reflected in this document. Approved by Mike Bare.	Released September 2022
8	Modifications to title page to reflect the latest product version supported. Approved by Kevin Chupp. All technical enhancements, changes, and fixes listed in the *Xpedition Enterprise Flow Release Notes* for this product are reflected in this document. Approved by Mike Bare.	Released March 2022
7	Modifications to title page to reflect the latest product version supported. Approved by Kevin Chupp. All technical enhancements, changes, and fixes listed in the *Xpedition Enterprise Flow Release Notes* for this product are reflected in this document. Approved by Mike Bare.	Released September 2021

Author: In-house procedures and working practices require multiple authors for documents. All associated authors for each topic within this document are tracked within the Siemens documentation source. For specific topic authors, contact the Siemens Digital Industries Software documentation department.

Revision History: Released documents include a revision history of up to four revisions. For earlier revision history, refer to earlier releases of documentation on Support Center.

Table of Contents

Chapter 1
Automation and Scripting....21
Automating Tasks through Scripting....21
Before You Begin....22
Compatible Scripting Languages....24
Running Scripts....25
Search Order for Scripts....25
The scripts.ini File....26
Debugging Scripts....28
Invalid Objects in Scripts....28
Specifying Script and Form Execution....29
Execution at Application Startup....29
Executing a Script or Form at Application Startup....29
Execution from the OS Command Window....30
Running a Script from the Command Line....30
Running a Form from the Command Line....30
Advantages of Using Forms....31
Key Binding....32
Key Binding Definition File....33

Chapter 2
Examples of Schematic Editor Automation....35
Script Examples in this Document....35
Example 1: Create/Add Menus....35
Example 2: Opening a Datasheet....37
Example 3: Using Objects in Scripts....38
Example 4: Using Applications and Documents in Scripts....45

Chapter 3
Schematic Editor Data Objects....51
AddinInfo Object....53
InitiallyDisabled Property (AddinInfo Object)....54
InitiallyVisible Property (AddinInfo Object)....55
LicenseFeature Property (AddinInfo Object)....56
Name Property (AddinInfo Object)....57
Placement Property (AddinInfo Object)....58
ProgId Property (AddinInfo Object)....59
RuntimeCreateDecision Property (AddinInfo Object)....60
ShortCutKey Property (AddinInfo Object)....61
ToolbarButton Property (AddinInfo Object)....62
Application Object....63
Activate Method (Application Object)....67
AddAddin Method (Application Object)....68
AppendOutput Method (Application Object)....69

CloseProject Method (Application Object)....71
CommandsManager Method (Application Object)....72
DesignComponents Method (Application Object)....73
DesignNets Method (Application Object)....74
DesignPaths Method (Application Object)....75
GetActiveDesign Method (Application Object)....76
GetDefaultColor Method (Application Object)....77
GetProjectData Method (Application Object)....78
Initialize Method (Application Object)....79
MGCDesignerDataManagement Method (Application Object)....80
NewProject Method (Application Object)....81
OpenBlocks Method (Application Object)....82
OpenProject Method (Application Object)....83
OpenURL Method (Application Object)....84
ParamGetMode Method (Application Object)....85
ParamGetValue Method (Application Object)....86
ParamSetMode Method (Application Object)....87
ParamSetValue Method (Application Object)....88
PrintProject Method (Application Object)....89
ProductionLibraryLimitationsChanged Method (Application Object)....90
PushPath Method (Application Object)....91
Query Method (Application Object)....93
QueryPages Method (Application Object)....94
Quit Method (Application Object)....95
RunISE Method (Application Object)....96
SchematicSheetDocuments Method (Application Object)....97
SelectPath Method (Application Object)....98
SelectPathCompPin Method (Application Object)....100
SetDefaultColor Method (Application Object)....102
SetRedraw Method (Application Object)....104
StartMigration Method (Application Object)....105
ActiveDocument Property (Application Object)....106
ActiveView Property (Application Object)....107
Addins Property (Application Object)....108
CommandBars Property (Application Object)....109
CommandLineArguments Property (Application Object)....111
Interactive Property (Application Object)....112
QueueSelectEvents Property (Application Object)....113
ShellCmd Property (Application Object)....114
SilentMode Property (Application Object)....115
SourceDocuments Property (Application Object)....116
StatusBarText Property (Application Object)....117
Version Property (Application Object)....118
Visible Property (Application Object)....119
ActivateView Event (Application Object)....120
ActivateView2 Event (Application Object)....121
AfterDocumentOpened Event (Application Object)....122
AfterPrintProject Event (Application Object)....123
AfterSheetRead Event (Application Object)....124

AfterSheetReRead Event (Application Object)....125
BeforeDocumentOpened Event (Application Object)....126
BeforePrintProject Event (Application Object)....127
BeforeProjectChanged Event (Application Object)....128
BlockLocked Event (Application Object)....129
BlockModified Event (Application Object)....130
CreateObject Event (Application Object)....131
DeactivateView Event (Application Object)....132
DeactivateView2 Event (Application Object)....133
Delete Event (Application Object)....134
DocumentClose Event (Application Object)....135
LockRequest Event (Application Object)....136
MouseMoved Event (Application Object)....137
PaintRegion Event (Application Object)....138
PrintFile Event (Application Object)....139
ProductionLibraryLimitationsChanged Event (Application Object)....140
ProjectChanged Event (Application Object)....141
ProjectClosed Event (Application Object)....142
Select Event (Application Object)....143
Shutdown Event (Application Object)....144
SourceDocumentSave Event (Application Object)....145
SourceFileModified Event (Application Object)....146
Startup Event (Application Object)....147
SymbolPreviewed Event (Application Object)....148
Unlock Event (Application Object)....149
Arc Object....150
GetLocation Method (Arc Object)....151
GetObjectColor Method (Arc Object)....152
IsColorAutomatic Method (Arc Object)....153
SetAutomaticColor Method (Arc Object)....154
SetLocation Method (Arc Object)....155
SetObjectColor Method (Arc Object)....156
Application Property (Arc Object)....157
LineStyle Property (Arc Object)....158
Parent Property (Arc Object)....159
Selected Property (Arc Object)....160
Type Property (Arc Object)....161
Attribute Object....162
Delete Method (Attribute Object)....164
DeleteInstanceValue Method (Attribute Object)....165
GetLocation Method (Attribute Object)....166
GetOatFull Method (Attribute Object)....167
GetObjectColor Method (Attribute Object)....168
IsColorAutomatic Method (Attribute Object)....169
SetAutomaticColor Method (Attribute Object)....170
SetLocation Method (Attribute Object)....171
SetObjectColor Method (Attribute Object)....172
Application Property (Attribute Object)....173
Child Property (Attribute Object)....174

EitherValue Property (Attribute Object)....175
Font Property (Attribute Object)....176
InstanceValue Property (Attribute Object)....177
Name Property (Attribute Object)....178
Orientation Property (Attribute Object)....179
Origin Property (Attribute Object)....180
Parent Property (Attribute Object)....181
Selected Property (Attribute Object)....182
Size Property (Attribute Object)....183
TextString Property (Attribute Object)....184
Type Property (Attribute Object)....185
Value Property (Attribute Object)....186
Visible Property (Attribute Object)....187
Block Object....188
AddArc Method (Block Object)....190
AddAttribute Method (Block Object)....191
AddBatchAttributes Method (Block Object)....192
AddBox Method (Block Object)....194
AddCircle Method (Block Object)....195
AddFub Method (Block Object)....196
AddLine Method (Block Object)....197
AddLine2 Method (Block Object)....199
AddNet Method (Block Object)....200
AddNetEx Method (Block Object)....202
AddPartInstance Method (Block Object)....204
AddPin Method (Block Object)....205
AddPinAtLocation Method (Block Object)....206
AddSymbolInstance Method (Block Object)....207
AddText Method (Block Object)....208
ApplySymbolUpdate Method (Block Object)....209
ChangeBorder Method (Block Object)....210
ChangeComponent Method (Block Object)....211
ChangeComponentPreserveRefdes Method (Block Object)....212
ClearHighlight Method (Block Object)....213
DeleteBorder Method (Block Object)....214
DeleteSelected Method (Block Object)....215
DeSelectAll Method (Block Object)....216
FindAttribute Method (Block Object)....217
GetBatchAttributes Method (Block Object)....218
GetBboxPoint Method (Block Object)....219
GetChildBlock Method (Block Object)....220
GetName Method (Block Object)....221
InsertBorder Method (Block Object)....222
PromoteSymbolNumbers Method (Block Object)....223
RepositionAttributesAsOnSymbol Method (Block Object)....224
SetZSheetSize Method (Block Object)....225
UpdateBorder Method (Block Object)....226
Application Property (Block Object)....227
Attributes Property (Block Object)....228

DataType Property (Block Object)....229
IsFub Property (Block Object)....230
LibraryName Property (Block Object)....231
OpenMode Property (Block Object)....232
Parent Property (Block Object)....233
SheetNum Property (Block Object)....234
SheetSize Property (Block Object)....235
SymbolType Property (Block Object)....236
Type Property (Block Object)....237
UpdateOutline Method (Block Object)....238
Box Object....239
GetLocation Method (Box Object)....240
GetObjectColor Method (Box Object)....241
GetObjectFillColor Method (Box Object)....242
IsColorAutomatic Method (Box Object)....243
IsFillColorAutomatic Method (Box Object)....244
SetAutomaticColor Method (Box Object)....245
SetAutomaticFillColor Method (Box Object)....246
SetLocation Method (Box Object)....247
SetObjectColor Method (Box Object)....248
SetObjectFillColor Method (Box Object)....249
Application Property (Box Object)....250
FillStyle Property (Box Object)....251
LineStyle Property (Box Object)....252
Parent Property (Box Object)....253
Selected Property (Box Object)....254
Type Property (Box Object)....255
CColor Object....256
b Property (CColor Object)....257
g Property (CColor Object)....258
r Property (CColor Object)....259
Circle Object....260
GetCenter Method (Circle Object)....261
GetObjectColor Method (Circle Object)....262
GetObjectFillColor Method (Circle Object)....263
IsColorAutomatic Method (Circle Object)....264
IsFillColorAutomatic Method (Circle Object)....265
SetAutomaticColor Method (Circle Object)....266
SetAutomaticFillColor Method (Circle Object)....267
SetCenter Method (Circle Object)....268
SetObjectColor Method (Circle Object)....269
SetObjectFillColor Method (Circle Object)....270
Application Property (Circle Object)....271
FillStyle Property (Circle Object)....272
LineStyle Property (Circle Object)....273
Parent Property (Circle Object)....274
Radius Property (Circle Object)....275
Selected Property (Circle Object)....276
Type Property (Circle Object)....277

CommandsManager Object..... 278
CommandDisable Method (CommandsManager Object).....279
CommandEnable Method (CommandsManager Object).....280
CommandRemove Method (CommandsManager Object)..... 281
ExecuteCommand Method (CommandsManager Object).....282
ExecuteMenuCommand Method (CommandsManager Object).....283
RegisterOLECommand Method (CommandsManager Object)..... 284
UnregisterOLECommand Method (CommandsManager Object)..... 286
Component Object..... 287
AddAttribute Method (Component Object).....289
AddBatchAttributes Method (Component Object).....290
AddBatchOats Method (Component Object)..... 292
AddLabel Method (Component Object)..... 294
AddOat Method (Component Object)..... 295
FindAttribute Method (Component Object).....296
GetBatchAttributes Method (Component Object)..... 297
GetBatchOats Method (Component Object).....298
GetBboxPoint Method (Component Object)..... 299
GetConnections Method (Component Object).....300
GetForwardPCB Method (Component Object)..... 301
GetLocation Method (Component Object).....302
GetName Method (Component Object).....303
SetLocation Method (Component Object)..... 304
Application Property (Component Object)..... 305
Attributes Property (Component Object).....306
Id Property (Component Object).....307
Label Property (Component Object).....308
Orientation Property (Component Object)..... 309
Parent Property (Component Object)..... 310
Refdes Property (Component Object)..... 311
Scale Property (Component Object).....312
Selected Property (Component Object).....313
SymbolBlock Property (Component Object).....314
Type Property (Component Object).....315
UID Property (Component Object)..... 316
ComponentPin Object..... 317
AddAttribute Method (ComponentPin Object)..... 318
AddOAT Method (ComponentPin Object).....319
FindAttribute Method (ComponentPin Object).....320
GetLocation Method (ComponentPin Object).....321
Application Property (ComponentPin Object).....322
Attributes Property (ComponentPin Object)..... 323
Component Property (ComponentPin Object).....324
Connection Property (ComponentPin Object)..... 325
Number Property (ComponentPin Object).....326
Parent Property (ComponentPin Object).....327
Pin Property (ComponentPin Object)..... 328
Selected Property (ComponentPin Object).....329
Side Property (ComponentPin Object)..... 330

Type Property (ComponentPin Object)....331
Connection Object....332
CompPin Property (Connection Object)....333
Net Property (Connection Object)....334
Ripper Property (Connection Object)....335
Segment Property (Connection Object)....336
DesignerDataManagement Object....337
AdditionalProperties Method (DesignerDataManagement Object)....338
Connect Method (DesignerDataManagement Object)....339
entities Method (DesignerDataManagement Object)....340
FindEntityByPath Method (DesignerDataManagement Object)....341
FindEntityByPathAndVersion Method (DesignerDataManagement Object)....342
IsConnected Method (DesignerDataManagement Object)....343
LastErrorMessage Method (DesignerDataManagement Object)....344
OpenComponentView Method (DesignerDataManagement Object)....345
OpenManagedBlockView Method (DesignerDataManagement Object)....346
RecentEntities Method (DesignerDataManagement Object)....347
ValidEntityTypes Method (DesignerDataManagement Object)....348
Option Property (DesignerDataManagement Object)....349
OptionIsReadOnly Property (DesignerDataManagement Object)....350
DesignerDataManagementAdditionalProperty Object....351
id Method (DesignerDataManagementAdditionalProperty Object)....352
Label Method (DesignerDataManagementAdditionalProperty Object)....353
DesignerDataManagementEntity Object....354
AccessType Method (DesignerDataManagementEntity Object)....356
AdditionalPropertyValue Method (DesignerDataManagementEntity Object)....357
CancelEditing Method (DesignerDataManagementEntity Object)....358
CheckedInDate Method (DesignerDataManagementEntity Object)....359
CheckIn Method (DesignerDataManagementEntity Object)....360
DumpProperties Method (DesignerDataManagementEntity Object)....361
EditShared Method (DesignerDataManagementEntity Object)....362
EditExclusive Method (DesignerDataManagementEntity Object)....363
Export Method (DesignerDataManagementEntity Object)....364
LaunchWebView Method (DesignerDataManagementEntity Object)....365
Name Method (DesignerDataManagementEntity Object)....366
OpenReference Method (DesignerDataManagementEntity Object)....367
Path Method (DesignerDataManagementEntity Object)....368
ProjectName Method (DesignerDataManagementEntity Object)....369
PropertyValueById Method (DesignerDataManagementEntity Object)....370
RelatedEntities Method (DesignerDataManagementEntity Object)....371
RunConfigurationRules Method (DesignerDataManagementEntity Object)....372
Status Method (DesignerDataManagementEntity Object)....373
Type Method (DesignerDataManagementEntity Object)....374
Unlock Method (DesignerDataManagementEntity Object)....375
Users Method (DesignerDataManagementEntity Object)....376
Version Method (DesignerDataManagementEntity Object)....377
View Method (DesignerDataManagementEntity Object)....378
ViewForOutputsGeneration Method (DesignerDataManagementEntity Object)....379
HDLSourceDocument Object....380

BookmarkLine Method (HDLSourceDocument Object)....381
GotoLine Method (HDLSourceDocument Object)....382
Name Property (HDLSourceDocument Object)....383
Path Property (HDLSourceDocument Object)....384
Label Object....385
GetLocation Method (Label Object)....386
GetObjectColor Method (Label Object)....387
IsColorAutomatic Method (Label Object)....388
SetAutomaticColor Method (Label Object)....389
SetLocation Method (Label Object)....390
SetObjectColor Method (Label Object)....391
Application Property (Label Object)....392
Font Property (Label Object)....393
Orientation Property (Label Object)....394
Origin Property (Label Object)....395
Parent Property (Label Object)....396
ResolvedName Property (Label Object)....397
Scope Property (Label Object)....398
Selected Property (Label Object)....399
Sense Property (Label Object)....400
Size Property (Label Object)....401
TextString Property (Label Object)....402
Type Property (Label Object)....403
Visible Property (Label Object)....404
Line Object....405
AddPoint Method (Line Object)....406
GetNumPoints Method (Line Object)....407
GetObjectColor Method (Line Object)....408
GetObjectFillColor Method (Line Object)....409
GetPoint Method (Line Object)....410
IsColorAutomatic Method (Line Object)....411
IsFillColorAutomatic Method (Line Object)....412
SetAutomaticColor Method (Line Object)....413
SetAutomaticFillColor Method (Line Object)....414
SetObjectColor Method (Line Object)....415
SetObjectFillColor Method (Line Object)....416
Application Property (Line Object)....417
LineStyle Property (Line Object)....418
Parent Property (Line Object)....419
Selected Property (Line Object)....420
Type Property (Line Object)....421
Net Object....422
AddAttribute Method (Net Object)....424
AddLabel Method (Net Object)....425
Connections Method (Net Object)....426
FindAttribute Method (Net Object)....427
GetConnectedLabel Method (Net Object)....428
GetConnectedNetName Method (Net Object)....429
GetLabel Method (Net Object)....430

GetObjectColor Method (Net Object)..... 431
GetRippers Method (Net Object)..... 432
GetSegments Method (Net Object)..... 433
GetSignals Method (Net Object)..... 434
GetSingleJointLocs Method (Net Object)..... 435
IsColorAutomatic Method (Net Object)..... 436
IsSegmentSelected Method (Net Object)..... 437
SelectSegment Method (Net Object)..... 438
SelectSegmentByJointLoc Method (Net Object)..... 439
SetAutomaticColor Method (Net Object)..... 440
SetObjectColor Method (Net Object)..... 441
Application Property (Net Object)..... 442
Attributes Property (Net Object)..... 443
Id Property (Net Object)..... 444
LineStyle Property (Net Object)..... 445
Parent Property (Net Object)..... 446
Selected Property (Net Object)..... 447
Type Property (Net Object)..... 448
UID Property (Net Object)..... 449
PDBPartitions Object..... 450
AppendPDBPartition Method (PDBPartitions Object)..... 451
GetPDBPartition Method (PDBPartitions Object)..... 452
GetPDBPartitionsArray Method (PDBPartitions Object)..... 453
InsertPDBPartition Method (PDBPartitions Object)..... 454
PDBPartitionExists Method (PDBPartitions Object)..... 455
RemovePDBPartitionByIndex Method (PDBPartitions Object)..... 456
RemovePDBPartitionByName Method (PDBPartitions Object)..... 457
Pin Object..... 458
AddAttribute Method (Pin Object)..... 459
FindAttribute Method (Pin Object)..... 460
GetLocation Method (Pin Object)..... 461
GetName Method (Pin Object)..... 462
GetObjectColor Method (Pin Object)..... 463
SetLocation Method (Pin Object)..... 464
Application Property (Pin Object)..... 465
Attributes Property (Pin Object)..... 466
Id Property (Pin Object)..... 467
Label Property (Pin Object)..... 468
Parent Property (Pin Object)..... 469
Selected Property (Pin Object)..... 470
Sense Property (Pin Object)..... 471
Side Property (Pin Object)..... 472
Type Property (Pin Object)..... 473
UID Property (Pin Object)..... 474
Point Object..... 475
X Property (Point Object)..... 476
Y Property (Point Object)..... 477
ProjectData Object..... 478
AddiCDBDesign Method (ProjectData Object)..... 480

GetBordersFilePath Method (ProjectData Object)....481
GetBusContentsFilePath Method (ProjectData Object)....482
GetiCDBDesignRootBlock Method (ProjectData Object).... 483
GetiCDBDesigns Method (ProjectData Object).... 484
GetiCDBDesignType Method (ProjectData Object).... 485
GetiCDBDiscardFilePath Method (ProjectData Object)....486
GetPCBDesignPath Method (ProjectData Object).... 487
GetPDBPartitions Method (ProjectData Object).... 488
GetPinComponentsFilePath Method (ProjectData Object)....489
GetProjectFilePath Method (ProjectData Object).... 490
GetProjectName Method (ProjectData Object)....491
GetProjectPath Method (ProjectData Object)....492
GetSearchPathScheme Method (ProjectData Object).... 493
GetSymbolPartitions Method (ProjectData Object).... 494
RemoveiCDBDesign Method (ProjectData Object).... 495
RenameiCDBDesign Method (ProjectData Object).... 496
SetBordersFilePath Method (ProjectData Object).... 497
SetBusContentsFilePath Method (ProjectData Object).... 498
SetiCDBDesignRootBlock Method (ProjectData Object)....499
SetiCDBDesignType Method (ProjectData Object).... 500
SetiCDBDiscardFilePath Method (ProjectData Object).... 501
SetPCBDesignPath Method (ProjectData Object).... 502
SetPinComponentsFilePath Method (ProjectData Object).... 503
SetSearchPathScheme Method (ProjectData Object).... 504
UpdateOtherObjects Method (ProjectData Object).... 505
CentralLibraryPath Property (ProjectData Object)....506
iCDBDir Property (ProjectData Object)....507
Rect Object....508
Bottom Property (Rect Object).... 509
Left Property (Rect Object)....510
Right Property (Rect Object).... 511
Top Property (Rect Object).... 512
Ripper Object.... 513
GetConnectedObject Method (Ripper Object)....514
GetConnectedObjects Method (Ripper Object).... 515
GetMappedSignal Method (Ripper Object)....516
SchematicSheetDocument Object.... 517
Activate Method (SchematicSheetDocument Object)....519
Close Method (SchematicSheetDocument Object).... 520
DiscardSymbolChanges Method (SchematicSheetDocument Object).... 521
ExportMetafile Method (SchematicSheetDocument Object)....522
GetViews Method (SchematicSheetDocument Object).... 523
IsReadOnly Method (SchematicSheetDocument Object)....524
Print Method (SchematicSheetDocument Object).... 525
ReRead Method (SchematicSheetDocument Object).... 526
Save Method (SchematicSheetDocument Object).... 527
SaveAs Method (SchematicSheetDocument Object).... 528
UpdateSymbolInDesign Method (SchematicSheetDocument Object)....529
Application Property (SchematicSheetDocument Object).... 530

FullName Property (SchematicSheetDocument Object)..... 531
Name Property (SchematicSheetDocument Object)..... 532
Parent Property (SchematicSheetDocument Object)..... 533
Segment Object.....534
GetJointType Method (Segment Object)..... 535
IsBus Method (Segment Object)..... 536
Location Method (Segment Object).....537
Application Property (Segment Object)..... 538
Attributes Property (Segment Object).....539
Parent Property (Segment Object)..... 540
Type Property (Segment Object)..... 541
ShellExecution Object..... 542
Open Method (ShellExecution Object)..... 543
PID Property (ShellExecution Object) 544
RedirectOutput Property (ShellExecution Object)..... 545
Run Method (ShellExecution Object)..... 546
Terminate Method (ShellExecution Object)..... 548
WorkingDirectory Property (ShellExecution Object)..... 549
Exit Event (ShellExecution Object).....550
Start Event (ShellExecution Object)..... 551
StdError Event (ShellExecution Object).....552
StdOutput Event (ShellExecution Object).....553
SymbolPartitions Object..... 554
AppendSymbolPartition Method (SymbolPartitions Object).....555
GetSymbolPartition Method (SymbolPartitions Object)..... 556
GetSymbolPartitionsArray Method (SymbolPartitions Object).....557
GetSymbolPartitionsCount Method (SymbolPartitions Object).....558
InsertSymbolPartition Method (SymbolPartitions Object).....559
RemoveSymbolPartitionByIndex Method (SymbolPartitions Object).....560
RemoveSymbolPartitionByName Method (SymbolPartitions Object).....561
SymbolPartitionExists Method (SymbolPartitions Object)..... 562
Text Object..... 563
GetLocation Method (Text Object).....564
GetObjectColor Method (Text Object)..... 565
IsColorAutomatic Method (Text Object).....566
SetAutomaticColor Method (Text Object)..... 567
SetLocation Method (Text Object)..... 568
SetObjectColor Method (Text Object).....569
Application Property (Text Object).....570
Font Property (Text Object)..... 571
Orientation Property (Text Object)..... 572
Origin Property (Text Object)..... 573
Parent Property (Text Object)..... 574
Selected Property (Text Object).....575
Size Property (Text Object).....576
TextString Property (Text Object).....577
Type Property (Text Object).....578
View Object..... 579
Activate Method (View Object)..... 581

AddAttributeMoveMode Method (View Object)....582
Application Method (View Object).... 583
BufferCopy Method (View Object).... 584
BufferCut Method (View Object)....585
BufferPaste Method (View Object).... 586
BufferPasteXY Method (View Object)....587
ComputeMBB Method (View Object)....588
Document Method (View Object)....589
GetJointLocs Method (View Object).... 590
GetName Method (View Object)....591
GetSelectedNetName Method (View Object).... 592
GetTopLevelDesignName Method (View Object).... 593
ModifyVisibility Method (View Object)....594
Query Method (View Object).... 595
Refresh Method (View Object).... 596
SelectbyName Method (View Object)....597
SelectbyName2 Method (View Object)....598
SelectObject Method (View Object)....599
SelectSegmentByJointLoc Method (View Object).... 601
SelectText Method (View Object).... 602
SetCenter Method (View Object)....603
ViewFull Method (View Object)....604
ZoomIn Method (View Object)....605
ZoomOut Method (View Object).... 606
ZoomSelect Method (View Object)....607
Block Property (View Object)....608
TopBlock Property (View Object)....609
Viewport Property (View Object)....610
OnActivate Event (View Object).... 611
OnSelect Event (View Object).... 612
Viewport Object.... 613
Arc Method (Viewport Object).... 615
Arrow Method (Viewport Object).... 616
Box Method (Viewport Object)....617
Circle Method (Viewport Object)....618
Ellipse Method (Viewport Object).... 619
EraseRectangle Method (Viewport Object).... 620
GetObjectColor Method (Viewport Object).... 621
Line Method (Viewport Object).... 622
PixelRectangle Method (Viewport Object)....623
PixelToUser Method (Viewport Object)....624
Point Method (Viewport Object)....625
PolyLine Method (Viewport Object).... 626
SetClipRectangle Method (Viewport Object).... 627
SetObjectColor Method (Viewport Object)....628
Spline Method (Viewport Object)....629
Text Method (Viewport Object).... 631
UserRectangle Method (Viewport Object).... 632
UserToPixel Method (Viewport Object)....633

FillStyle Property (Viewport Object)....634
LineCap Property (Viewport Object)....635
LineJoin Property (Viewport Object)....636
LinePattern Property (Viewport Object)....637
LineThickness Property (Viewport Object)....638
RasterMode Property (Viewport Object)....639
TextAngle Property (Viewport Object)....640
TextFont Property (Viewport Object)....641
TextSize Property (Viewport Object)....642

Chapter 4
Schematic Editor Object Collections....643
DesignerDataManagementAdditionalProperties Collection....644
Count Method (DesignerDataManagementAdditionalProperties Collection)....645
Item Method (DesignerDataManagementAdditionalProperties Collection)....646
Remove Method (DesignerDataManagementAdditionalProperties Collection)....647
DesignerDataManagementEntities Collection....648
Count Method (DesignerDataManagementEntities Collection)....649
Item Method (DesignerDataManagementEntities Collection)....650
Remove Method (DesignerDataManagementEntities Collection)....651
HDLSourceDocuments Collection....652
Item Method (HDLSourceDocuments Collection)....653
New Method (HDLSourceDocuments Collection)....654
Open Method (HDLSourceDocuments Collection)....655
Remove Method (HDLSourceDocuments Collection)....656
RemoveAll Method (HDLSourceDocuments Collection)....657
SaveAll Method (HDLSourceDocuments Collection)....658
Count Property (HDLSourceDocuments Collection)....659
SchematicSheetDocuments Collection....660
Close Method (SchematicSheetDocuments Collection)....662
CopyToClipboard Method (SchematicSheetDocuments Collection)....663
DeleteSheet Method (SchematicSheetDocuments Collection)....666
GetAvailableSchematics Method (SchematicSheetDocuments Collection)....667
GetAvailableSheets Method (SchematicSheetDocuments Collection)....668
InsertSheet Method (SchematicSheetDocuments Collection)....669
IsSymbolUnderEdit Method (SchematicSheetDocuments Collection)....670
Item Method (SchematicSheetDocuments Collection)....671
Open Method (SchematicSheetDocuments Collection)....672
Open_Hierarchically Method (SchematicSheetDocuments Collection)....673
OpenSymbol Method (SchematicSheetDocuments Collection)....674
PasteFromClipboard Method (SchematicSheetDocuments Collection)....675
Application Property (SchematicSheetDocuments Collection)....676
Count Property (SchematicSheetDocuments Collection)....677
Parent Property (SchematicSheetDocuments Collection)....678
StringCollection Collection....679
Item Method (StringCollection Collection)....680
Remove Method (StringCollection Collection)....681
Count Property (StringCollection Collection)....682

StringList Collection....683
Append Method (StringList Collection)....684
Clear Method (StringList Collection)....685
GetCount Method (StringList Collection)....686
GetItem Method (StringList Collection)....687
Insert Method (StringList Collection)....688
Remove Method (StringList Collection)....689

Chapter 5
Schematic Editor Enumerated Types....691
Schematic Editor Enumerated Types Summary....692
DesignerErrCode Enum....694
PinMappingType Enum....696
PropertyMappingType Enum....697
ScopeReplaceSymbol Enum....698
ShellEventID Enum....699
ToolType Enum....700
VdAllOrSelected Enum....701
VdAnnoObject Enum....702
VdAnnoPos Enum....703
VdAppEventDispatchID Enum....704
VdArcPoint Enum....707
VdArrowType Enum....708
VdBoolean Enum....709
VdBusOrWire Enum....710
VdCompInstanceForwardPCB Enum....711
VdCorner Enum....712
VdCreateTime Enum....713
VdDataType Enum....714
VdDMDumpPropertiesFilter Enum....715
VdDMEntityAccessType Enum....716
VdDMEntityType Enum....717
VdDocumentAccess Enum....718
VdFillStyle Enum....719
VdFont Enum....721
VdGradientType Enum....722
VdJointType Enum....723
VdLabelVisibility Enum....724
VdLineCap Enum....725
VdLineJoin Enum....726
VdLinePattern Enum....727
VdLineStyle Enum....728
VdNameType Enum....729
VdNotifyFlag Enum....730
VdObjectClass Enum....732
VdObjectType Enum....733
VdObjectTypeMask Enum....735
VdOnOff Enum....737

VdOpenMode Enum....738
VdOrientation Enum....739
VdOrigin Enum....740
VdParamMode Enum....741
VdParamValue Enum....745
VdPEFlowMode Enum....748
VdPinEndType Enum....749
VdRasterop Enum....750
VdScope Enum....751
VdSegmentEndType Enum....752
VdSelectionType Enum....753
VdSense Enum....754
VdSheetSize Enum....755
VdSide Enum....757
VdSilentMode Enum....758
VdSourceDocumentType Enum....759
VdSplineOrder Enum....760
VdSplineType Enum....761
VdSymbolType Enum....762
VdTextFlags Enum....763
VdUpdateOOScope Enum....764
VdUpdateOtherObjects Enum....765
VdVisibilityFlag Enum....766
VdWhichJoint Enum....767
WindowStyles Enum....768

Chapter 6
Scripting with Databook....769
Databook Objects....770
Attribute Object....771
Name Property (Attribute Object)....772
NameVisible Property (Attribute Object)....773
Value Property (Attribute Object)....774
ValueVisible Property (Attribute Object)....775
Attributes Object....776
Add Method (Attributes Object)....777
Count Property (Attributes Object)....778
Component Object....779
Attributes Property (Component Object)....780
Instances Property (Component Object)....781
Library Property (Component Object)....782
Properties Property (Component Object)....783
Symbol Property (Component Object)....784
Components Object....785
Count Property (Components Object)....786
Item Method (Components Object)....787
Application_AddComponent Event....788
Application_AfterAddComponent Event....790

Application_AfterAnnotateComponent Event....792
Application_AnnotateComponent Event....794
Application_LoadComponent Event....796
Property Object....798
Item Method (Property Object)....799
Remove Method (Property Object)....800
Application_SelectComponent Event....801
Application_ViewDocument Event....802
Configuring the Databook Script....803

Appendix A
Changes to Schematic Editor Automation....805
Changes to Objects....806
Removed Objects....806
Changes to the Application Object....806
Changes to the Block Object....809
Changes to the Arc Object....810
Changes to the Attribute Object....810
Changes to the Box Object....811
Changes to the Circle Object....811
Changes to the Component Object....811
Changes to the Connection Object....812
Changes to the Label Object....812
Changes to the Line Object....813
Changes to the Net Object....813
Changes to the Pin Object....814
Changes to the Ripper Object....814
Changes to the Text Object....814
Changes to the Viewport Object....815
Changes to Enumerated Types....816
Removed Enumerated Types....816
Changes to the VdAppEventDispatchID Enumerated Type....816
Changes to the VdDocumentAccess Enumerated Type....817
Changes to the VdNotifyFlag Enumerated Type....818
Changes to the VdObjectClass Enumerated Type....819
Changes to the VdObjectType Enumerated Type....819
Changes to the VdObjectTypeMask Enumerated Type....820

Third-Party Information

Chapter 1
Automation and Scripting

Xpedition Designer gives you the ability to programmatically complete tasks using scripts. These customizations work seamlessly within the framework of Xpedition Designer, enhancing and leveraging its functionality to suit your exact requirements.

Scripting is a way to manipulate properties on objects in your schematic design, create custom design rule checks, generate reports (as message boxes or text files), control the user interface, and perform various other tasks. In short, you can use scripting to perform any action available in the Xpedition Designer application.

At its simplest, you can use scripting to combine commands to perform repetitive operations quickly and easily. However, by using the event-driven object model available in Xpedition Designer, you can build complete custom solutions, including new user interfaces.

The purpose of this document is to supply the information you need to get started using scripts with Xpedition Designer.

Automating Tasks through Scripting
Before You Begin
Compatible Scripting Languages
Running Scripts
Key Binding

Automating Tasks through Scripting

A script is a series of automation instructions that you group together as a single command to accomplish a task automatically. You can use scripting to automate time-consuming, repetitive operations in the application and you can also create scripts and forms that provide brand-new functionality, such as design-navigation aids and web browsing add-ins.

Some typical uses for scripts are:

- To automate a complex series of tasks
- To perform custom checks on a schematic
- To add a new command to a popup menu
- To add a new menu with custom commands that invoke user scripts or command line commands
- To build a new dialog box that adds functionality

Note:
In addition to the data models discussed in this document, Xpedition Designer makes use of the command bar server data model. Xpedition Designer accesses the methods and objects of the CommandBarServer via the application: Application.CommandBars

Perhaps more to the point, you can create scripts that will assist you as you create and test your design. Scripts can:

- Provide message boxes that inform you of the status of your project or design
- Generate reports, statistics, bills-of-materials, parts lists and other useful data regarding your design
- Automate the steps involved with changing design data, adding documentation to the design, or even adding components, nets or other items to the design
- Define custom design rule checks

Before You Begin

A script communicates with its host and other applications through a COM technology called Automation (formerly OLE Automation). To support automation, an application requires an object model that exposes certain objects, with their properties and methods, to external applications. The automation client can do anything which is supported by the automation server, including the extraction of information from the server, as well as object and property manipulation.

In this model the schematic application is the server, and the scripts and forms that you build are the clients. The relationship is conceptually illustrated in Figure 1: Server-client relationship in scripting on page 22.

Figure 1. Server-client relationship in scripting

With automation, the schematic application and its components become objects you can control programmatically. In addition to manipulating objects exposed by the scripting host, a script can also manipulate objects served by other automation servers. For example, a script in the schematic application can manipulate an Excel spreadsheet or a Word document. The reverse of this is also possible: a Word or Excel script can manipulate the schematic document.

In order to create effective script, it is necessary to understand the elements available for manipulation with scripting. These are objects, properties, methods, and events.

Objects are elements within an application that can be examined or manipulated by scripts, while properties are descriptive data that define the object with which they are associated. For example, a data model in Xpedition Designer might include the objects show in Figure 2: An Example Data Model on page 23.

Figure 2. An Example Data Model

Methods are operations that can be performed on an object. For example, methods can include closing or saving a document, or setting or retrieving values from an object. Events are actions that occur within the

application for which a script is written to respond. For example, a script may be triggered before or after a project document is opened or saved.

For information about working with events, refer to Working with Events in the Layout Automation Reference manual.

Compatible Scripting Languages

Scripting languages that Xpedition Designer supports include, but are not limited to, VBScript and JScript. Xpedition Designer (for the PC) is an ActiveX® Scripting host, meaning it can generally make use of any ActiveX scripting engine that you have installed.

There are a myriad of books and web sites dedicated to helping you learn good practices and techniques for script programming. The resources you choose to help you learn are largely a function of the scripting language you are using and the platform from which you operate. Rather than attempt to list all these resources in this space, Mentor Graphics recommends that you search the internet for an extensive list of the many resources available.

Note:
Note that many automation programmers that use the Windows platform rely on the Windows Script Documentation available from the Microsoft web site. There is online help at that site, available for download.

Running Scripts

You can specify that a script be executed during application startup (typically, adding or updating menus or toolbars), run it from the command line (using the “run” command, or you can bind it to a toolbar button, menu command or key-binding. You can also create forms with active elements (for example, a button) that call subroutines within the forms, scripts, batch files, or executables.

After you've assigned a script to a toolbar, menu or form element, running the script is as simple as clicking that toolbar button, menu item, key-binding, or form element.

Note:
Throughout this document, a script is defined as a code sequence which can be executed. A form is defined as a custom dialog box created by the user, or on the user’s behalf, with which the user interacts. The graphical elements (for example, buttons) in the dialog box have script subroutines associated with them to execute specific tasks.

Included in your application installation are samples of scripts and forms that you can test with Xpedition Designer.

Search Order for Scripts
The scripts.ini File
Debugging Scripts
Specifying Script and Form Execution

Search Order for Scripts

Whenever there is a call to execute a script, whether during startup, through the use of a form or command, or due to an event, the application uses a particular search order to locate the script file.

The search order that the application uses is as follows:

1. Directory in which the project file is located.
2. All directories specified by the WDIR environment variable

For a script to be considered for execution by the application, it must be located in one of the directories above, or the path and file name must be specified in the command line (see “Running a Script from the Command Line” on page 30, and “Running a Form from the Command Line” on page 30).

The scripts.ini File

Text file that lists the full path to, and name of, the script files available to the application. You can specify that scripts be run at startup by including them in the scripts.ini file.

Description

You do not need to specify the full path to the script files if they exist in one of the default directories listed in Search Order for Scripts. These scripts and forms often contain menu or command customizations or custom process integration forms. These are typically files with *.vbs* or *.efm* extensions. Additionally, batch and executable files (*.bat* and *.exe*, respectively) may be executed on the Windows platforms. In Unix or Linux operating systems, file with execute permissions may be specified.

Multiple scripts.ini files may exist in different directories specified by the WDIR environment variable, in which case the scripts named within are executed according to the search order.

Note:
If there are multiple scripts with identical names in the various searched directories, only the first such script that the application encounters will be executed.

Format

When you develop a scripts.ini file, keep in mind that the following special characters have particular significance:

Table 1. Special characters scripts.ini Files

[*application*]	Square brackets must be used to enclose the name of the application to which the scripts apply.
*	The asterisk is a wild card. When a script is located via a search that uses a wild card, the search stops. Therefore the wild card cannot be used to launch multiple scripts with similar names.
;	The semicolon is the comment character. Any text that follows a semicolon is considered comment text until a new line is located.
%*variable_name*% $*variable_name*	Environmental variables can be used in scripts. For example, $HOME can be used to identify the home directory.

Note:
Script numbering in the scripts.ini file must be sequential. If scripts are numbered non-sequentially, the first non-sequential script that is encountered, and all that follow it, are ignored.

Parameters

None

Examples

The following is an example of a scripts.ini file:

```
[Viewdraw]
script#0=MyMenuByName.vbs
;  This line is an example of a comment.
script#1=ToolBar.vbs
script#2=AddCorpMenus.vbs
script#4=BadExample.vbs    ;  This entry, and all the following are ignored
script#5=Ignored.vbs       ;     because script#3 is missing.
```

Debugging Scripts

A good debugger provides a debugging environment that extends an ActiveX Scripting host application.

There are many third party script debuggers available. A quick search of the internet using the keywords “script debugger” will reveal several.

The debbugger you choose should enable you to do the following:

- View the source code of the script you are debugging.
- Control the pace of script execution with break points and stepping.
- View and change variable and property values with the Command Window.
- View and control script flow with the Call Stack Window.
- View the script objects of the targeted applications.

Invalid Objects in Scripts

Invalid Objects in Scripts

Unless your script performs its own error handling, attempting to use an invalid object will cause the script to stop. In some cases, this could result in a run-time error report.

For example:

Object required: <*offending object name*>

In other cases, this may cause an application crash. For example, if you attempt to use an ActiveView object for a schematic sheet after the schematic sheet has been closed.

A good practice is to have the script check the validity of an object before using it. Following this practice also helps to diagnose problems with scripts because using invalid objects is a common cause of scripts not running to completion.

In this example, <*object Name*> is the name of an object in a script:

```
' Enable error-handling.
On Error Resume Next
If <object name> Is Nothing Then
 MsgBox <object name> & " Object not found."

Else
 MsgBox <object name> & " Object is valid."
End If
```

If necessary, use the On Error GoTo 0 statement to disable error handling by the script. For example:

```
‘Disable error-handling
On Error GoTo 0
```

Specifying Script and Form Execution

You can also run scripts by opening them from within these applications. You can write scripts to be self-contained custom commands or you can write script code to be executed through forms.

Forms are interfaces (windows, including dialogs) that you design by adding controls to a form and writing the script code to handle events associated with each control.

Generally, scripts run and then exit immediately. They do not wait for events, unless you specifically program an event “handler” into them. If you want a particular subroutine to run in response to an event, you must include the appropriate script code as part of a form, or provide the appropriate code within your script.

Execution at Application Startup
Executing a Script or Form at Application Startup
Execution from the OS Command Window
Running a Script from the Command Line
Running a Form from the Command Line
Advantages of Using Forms

Execution at Application Startup

Any scripts that are specified in the scripts.ini file are executed at startup.

For more information, refer to The scripts.ini File.

Executing a Script or Form at Application Startup

You can execute a script or form when you open the schematic application.

Note:
Be sure to select script (*.vbs*) files in the Scripts area, and form (*.efm*) files in the Forms area. Otherwise you may receive an error message.

Linux:
Do not use the Application.Quit method in scripts that run at startup. Application.Quit will cause an error when called before the Viewdraw application has fully initialized.

Procedure

1. In Xpedition Designer, select **Setup > Settings**.
2. Select the **Run on Startup** entry in the left hand panel.
3. Click the **New** button in the Script or Form group – or simply double-click the last line in the group.
4. Browse to and select the *.vbs* or *.efm* file that you want to load and click **Open.**

5. Click **OK** to dismiss the dialog box. Each time you start Xpedition Designer, the script or form runs.

   To change the order of execution of scripts or forms:

   - Select a *.vbs* or *.efm* file and click the Move Up or Move Down button.

   To remove a script or form:

   - Select a *.vbs* or *.efm* file and click the Delete button.

Note:
When you use **Run on Startup** to add, delete, or change the execution order of scripts or forms, the schematic application modifies the *scripts.ini* file accordingly.

Execution from the OS Command Window

You can run scripts and forms from the operating system's command prompt.

Note:
The Windows operating system has scripting hosts available. However, to assure that your scripts will work on all platforms, it is best to use the mgcscript scripting host to run a script client as follows (the mgcscript host will also run script clients written in JScript):

mgcscript MyClient.vbs *[parameters]*

Running a Script from the Command Line

You can run scripts from the Xpedition Designer command line.

Procedure

1. Enter **run <scriptname>** in the command-line entry toolbar.
2. Press the <ENTER> key or click the **Execute Command** button. The application runs the script you specify.

Running a Form from the Command Line

You can run forms from the Xpedition Designer command line.

Procedure

1. Enter **form *<formname>*** in the command-line entry toolbar.
2. Press the ENTER key or click the **Execute Command** button. The application opens and runs the form you specify.

Advantages of Using Forms

Some applications allow you to create forms within the application to launch scripts. Using a form gives you a visual front end for each script.

Once invoked, forms remain open (and running) until you switch to edit mode or close the form.

When you act on the form’s controls, as when selecting an item in a listbox or clicking a button, the appropriate functions in the form's script execute. Functions also execute for other reasons (as when the form loads or because of events fired by other objects). Each script can contain many subroutines or functions

In Xpedition Designer, for example, you could create a form that asks for an attribute name and visibility setting. Once you input the data in the form, it could run code that performs some action on all attributes that match the search criteria.

Key Binding

Xpedition Designer includes key bindings, specified in a script, that allow for certain keys (on the keyboard) to execute specific commands.

For example, the lower case “n” is mapped to the Add Net command. You can also map keys to execute commands such as “run,” to launch a form or execute a script from within the application.

Key Binding Definition File

Key Binding Definition File

Key binding is governed by definition files located in the project directory and any directories specified by the WDIR environment variable.

Description

The key binding file used depends on the setting selected in the Advanced tab of the Settings dialog box. When the Xpedition Layout Style Keybindings is enabled, the exped_wvo.vbs or the exped_pv.vbs files (located in %SDD_HOME%\standard\) are used, depending on the operating system. When it is disabled, the application will use the settings specified in the vdbindings.vbs file. See Table Table 2: Key Binding Definition File Usage

Table 2. Key Binding Definition File Usage

Operating System	Keybindings Enabled	Keybindings not Enabled
Windows	exped_wvo.vbs	vdbindings.vbs
UNIX/LINUX	exped_pv.vbs	vdbindings.vbs

Note:
These file names and locations may be overridden using a scripts.ini file. The scripts.ini file is executed during Xpedition Designer initialization only. This file must be located in the project - for project specific keybinding and stroke mapping - or in any of the directories specified by the WDIR environment variable for all projects. See Search Order for Scripts.

Format

You can edit these files to bind your scripts to various key combinations. Shortcut keys for scripts can be any of the following:

Enter, Space, A-Z, 0-9, and F1-F12

These keys can be modified with the **Alt**, **Ctrl**, and **Shift** keys (or the various combinations thereof).

Note:
If you use the F1-F12 keys to bind your scripts, include a modifier (**Alt**, **Ctrl**, or **Shift**). The application uses F1-F12 to provide functionality by default. If you bind these keys to your script without a key modifier, you will disable the application functionality.

Parameters

None

Examples

This example shows some typical key bindings.

```
' Constant Definition
Menu = 0  ' Used to specify
Cmd = 1
Accelerator = 1
Key = 2
NotSticky = 0
Sticky = 1

'  Bind a menu command to a key

' Bind the Xpedition Designer Application FileOpen menu selection to
' Ctrl+O

Bindings("Application").AddKeyBinding "Ctrl+O","FileOpen",Menu,Accelerator

' Bind Xpedition Designer Schematic Edit>Delete Special selection to
' the Delete key
Bindings("Schematic").AddKeyBinding "Delete", "EditDeleteSpecial", Menu,
 Key

' Bind the Xpedition Designer Stroke sequence 357 to the internal
' zoom in command
Bindings("Stroke").AddStroke "357", "WVOZoomIn"

' Bind the execution of a user script to a key
Bindings("Schematic").AddKeyBinding "h", "run c:\testkeybind.vbs", cmd,
 Key
```

Chapter 2
Examples of Schematic Editor Automation

The section provides working examples of Xpedition Designer automation. Many of the scripts in this section were written by users in the field. Refer to the scripting code used in these examples to help understand how you can use Xpedition Designer data objects, methods, properties, and events to automate Xpedition Designer tasks that you typically perform manually.

For details on managing scripts in an environment with multiple software installs see Scripting With Multiple Installs.

Script Examples in this Document
Example 1: Create/Add Menus
Example 2: Opening a Datasheet
Example 3: Using Objects in Scripts
Example 4: Using Applications and Documents in Scripts

Script Examples in this Document

Throughout this documentation, there are frequently code examples that illustrate various concepts.

Mentor Graphics recommends using COM versioning syntax where applicable. For more information, see “Identifying the COM Version Number of an Install” in *Common Automation Reference*.

Example 1: Create/Add Menus

This script adds a menu, a submenu, and menu items, to the application menu bar and Component popup menus and associates a script function with the selection of the menu item.

You can use this script via any of the methods described in “Running Scripts” on page 25.

Note:
Whenever you create a custom menu with a script, be sure to include the following line in the script (This line is necessary to prevent Designer from crashing when the script is executed):

```
Scripting.DontExit = True
```

For this script to run and for the created menu to appear, a schematic sheet must be active.

Note:
Xpedition Layout and Xpedition Designer each use different command bar servers. Scripts must call the correct command bar server for the respective product. Use MGCSDD.CommandBarsEx for Xpedition Layout menus; use CommandBarsEx by referring to CommandBars property for Xpedition Designer menus.

```
Scripting.DontExit = True
' Insert a new 6th menu in Sheet Menu Bar. When the sheet is opened, the
' the menu will appear between the existing Add and Format menus.
Set menu = CommandBars("Sheet Menu Bar").Controls.Add(cmdControlPopup,,,6)
' Menu text
menu.Caption="&Datasheet"
' Add a menu item to the menu
Set button = menu.Controls.Add
' Menu item text
button.Caption="View Datasheet"
' Action to perform when the menu item is selected.
button.OnAction="run Doc_example2.vbs"
' Menu item is "grayed" out and cannot be activated; Set to True
' to enable the menu item's selection.
button.Enabled = False
' *********
' Add a menu item to the Component Popup menu.  The menu item will
' open the datasheet for the selected component.
Set menu = CommandBars("Component Popup").Controls.Add(cmdControlPopup,,,2)
' Menu text
menu.Caption="&Datasheet"
' Add a submenu
Set button = menu.Controls.Add  ' Add a button to the menu
' Submenu item text
button.Caption="View Datasheet"
' Action to perform when the submenu item is selected
button.OnAction="run doc_example2.vbs"
'******
' Add an entry to an existing toolbar menu, View, to view the datasheet
' of the selected component.  The Item property is an Indexed property
' therefore, one must count the menus to identify index.  Be sure to
' include any "spacers" which may have been added.
'
' Modmenu represents the View menu Object.
Set Modmenu = CommandBars("Sheet Menu Bar").Controls.Item(3)
' Adding a menu item to the Modmenu Object, the View menu.
Set AddItem = Modmenu.controls.Add
' Menu item text
AddItem.Caption="View Datasheet"
' Action to perform when the menu item is selected
AddItem.OnAction="run doc_example2.vbs"
'
' ***********
' Example of deleting a menu item.
' Create an object which represents the menu, or menu item,
' to delete. If the menu is within a main menu, you need to
' "drill down" to that menu item, much the way the add works
' above.
Set DelmenuItem = CommandBars("Sheet Menu Bar").Controls.Item(8)
' "Execute" the Delete method.
DelMenuItem.Delete
```

Note:
In addition to the data models discussed in this document, Xpedition Designer makes use of the command bar server data model. Xpedition Designer accesses the methods and objects of the CommandBarServer via the application: Application.CommandBar

Example 2: Opening a Datasheet

The following script provides a flexible launching method for datasheets with an attribute populated with the filename in the Databook tool.

This example launches Adobe Acrobat Reader to open the datasheet PDF file. This script provides the functionality for the menu item created in "Example 1: Create/Add Menus" on page 35.

The script acquires the value of an environment variable, MY_DS. This value is the directory in which the datasheet files are located. The script then finds the DATA_SHEET property value of the selected component and launches the datasheet in Acrobat Reader.

In order for this script to run, a component must be selected when you choose the menu item.

You can use this script via any of the methods described in "Running Scripts" on page 25.

Note:
Mentor Graphics recommends that you use COM versioning syntax in script examples that use GetObject and CreateObject. Without COM versioning, the script will access the last installation to which the release switcher pointed.

```
' Get the attribute for DATA_SHEET from the part on the schematic
Option Explicit
' Used to populate  Object browser in script editor
' set vdapp = CreateObject("Viewdraw.Application")


' Run on an open document.
Dim vdapp,vddoc,vdview

' **** Use the code within the asterisk area for an open drawing ****
 Set vdapp = GetObject (,"ViewDraw.Application")
Scripting.AddTypeLibrary("ViewDraw.Application")
Set vddoc = vdapp.ActiveDocument
 Set vdview = vdapp.ActiveView
' ************* End of open drawing code ********************

' Get attributes on selected part
 Dim DataSheetDir, DataSheet, DataSheetName, RefDes, RefDesValue, Comp
 ' Get the value a system environment variable whose value is the
 ' directory containing the datasheets and advise the user.
 DataSheetDir = Scripting.GetEnvVariable ("MY_DS")

 MsgBox "DataSheetDir is " & DataSheetDir,,"Datasheet Directory"

 'Create a collection using the Query method, of the
```

```
'components which are selected on the schematic
For Each Comp in vdview.Query(VDM_COMP, VD_Selected)

 ' Identify the DATA_SHEET attribute on the component
 Set DataSheet = Comp.FindAttribute("DATA_SHEET")

 ' Use the If conditional to determine if the attribute exists.
 ' If it doesn't, advise the user.
 ' If it does, execute the statements after Else

  If DataSheet Is Nothing Then
   MsgBox "No DATA_SHEET Attribute on this part",,"Datasheet Exists?"
  Else
   ' Get the DATA_SHEET attribute value
   DataSheetName = DataSheet.value

   ' Processing the REFDES is optional and is used basically for
   ' completeness/debug.
   ' Create an object for the REFDES attribute
   Set RefDes = Comp.FindAttribute("Ref Designator")
   If RefDes Is Nothing Then
    MsgBox "No Ref Designator Property found.",,"What's the problem?"
   else
    RefDesValue = RefDes.value
   End If ' REFDES is nothing
   ' Check to see if the DATA_SHEET attribute has a value.
   If DataSheetName="" Then
    MsgBox "Datasheet name not specified for component " &
RefDesValue,,"Datasheet?"
   Else
    MsgBox "Refdes is: " & RefDesValue & vbCrLf &_
    "Datasheet filename is " & DataSheetName,,"Confirmation"
' Concatenate a defined directory path to the datasheet attribute value
    Dim DataSheetFileName
    DataSheetFileName = DataSheetDir & "\" & DataSheetName
    ' Inform the user of the datasheet which will be opened.
    MsgBox "Datasheet file Name: " & DataSheetFileName,, "Datasheet To
Open"

 ' Example of how to launch a third party executable
 Dim Win
 Set win = CreateObject("WScript.shell") ' Create the windows object
'Use the run method to launch an application on the specific datasheet
       win.run "Acrobat.exe " & DataSheetFileName
      End If
    End If
Next
```

Example 3: Using Objects in Scripts

The script in this section demonstrates how to interact with various objects.

The script demonstrates how to:

- Use the CreateObject method to launch an application.
- Check for a running process.
- Check for an open document.
- Use the Input Box.
- Open, write to, and close a text file.
- Open and write to an Excel file.
- Access component properties.
- Identify the block type of a component symbol.
- Select a component.
- Execute a command line command (zooms into the selected component).
- Deselect all objects on a page.
- Acquire the project directory.
- Create a collection of component objects.
- Launch a Windows based executable on a file.

You can use this script via any of the methods described in "Running Scripts" on page 25.

Note:
Mentor Graphics recommends that you use COM versioning syntax in script examples that use GetObject and CreateObject. Without COM versioning, the script will access the last installation to which the release switcher pointed.

```
Option Explicit

Dim vdapp, vddoc, vddocName, vdview, CurBlock, CurProjDir, Application
Dim OpenMySchematic, BadEntry, OpenSchematic,
 OpenSchematicCount,UboundOpenSch
Dim fso, CompRefXref
Dim CompRefdes, CompColl, CompObj, CompSymType, RefDesAttr, RefDesVal
Dim CompXLoc, CompYLoc, RefDesName, RefDesInstValue, RefDesSymbValue
Dim NoAppOpen, NoDocOpen
Dim VDM_COMP, VD_ALL, VDDT_SCHEMATIC
Dim CompCollCount, ProcCompCount
```

```
' Initializing counting variables
ProcCompCount = 0

' Seting up variables that will be used while checking for a running
' application, available document, and a prompt for the user for a
' schematic name.
 NoAppOpen = True
 NoDocOpen = True
 BadEntry = True

'
' The following constants are known within Xpedition Designer and
' DO NOT need to be defined when the script is run from within
' Xpedition Designer.  Running the script as a client using a debugger,
' Xpedition Designer doesn't understand it's'
' internal enumerated variables
' and their values, so we are providing them here.
'
' Once a script has been debugged, the intention is to run them
' within the Xpedition Designer application, it is not necessary to set
' the enumerations as they will be provided within the application.
'
 VDM_COMP =  128
 VD_ALL = 0
 VDDT_SCHEMATIC = 0

' Set script to continue if an error occurs.  This is required because
' there is "error trapping" code and we need the script, which will stop
' on an error, to continue to the error trapping code.
 On Error Resume Next
' Check to see if the application is running.  If it isn't
' prompt the user for a schematic name.
 Set Application = GetObject(,"ViewDraw.Application")
 If IsEmpty(Application)  Then
  MsgBox "Application is NOT running." & vbCrLf &_
      "It will be launched after you click OK.",,"Empty Check"
'  Launch the Xpedition Designer Application for X-ENTP VX.1 (COM Version
' 9)
  Set vdapp = CreateObject("Viewdraw.Application.9")
'  Set the application to visible (e.g. GUI is displayed, else the
' application runs in the "background".
     vdapp.Visible = True
'  The OpenProject method is an Application method.  The msgbox has been
'  placed here to allow the application to launch so that the Application '
 method may be executed.
     MsgBox "Wait for Xpedition Designer to Open",,"Pause"
' For the purpose of this example, the OpenProject method is hard coded.
' In actual application, the user may be prompted for the project file.
     vdapp.OpenProject
 ("c:\MGTraining2007.1\solutions\DXD2007\DXD2007.prj")
'     Scripting.AddTypeLibrary("Viewdraw.Application")
   ' Prompt the user for a schematic name.  Loop on the dialog if
```

```
   ' they don't enter a name, or simply accept the default.
While NoAppOpen
If Not OpenMySchematic="NameRequired" Or Not OpenMySchematic="" Then
' Prompt user for the desired schematic name and page number.
OpenMySchematic = InputBox ("Schematic Name, Page Number (e.g. Top,1): ",_
         "User Prompt","NameRequired",_
          500, 500)
 ' Split the user input so the user input can be put in the appropriate
 ' format for the SchematicSheetDocuments.Open method below.
 OpenSchematic = Split (OpenMySchematic,",",-1)
 ' Use the UBound function to get the upper limit of the resulting
 ' array.  This could also be used with some code as an "error"
 ' check (e.g. did the user enter the right number of strings?, etc.)
      UboundOpenSch = UBound (OpenSchematic)
 ' The array generated by the Split function is a zero reference array.
 ' So, the actual quantity of elements is the UBound value plus 1.
      OpenSchematicCount = UboundOpenSch + 1
 ' A message box is a good way of displaying the data to ensure that it
 ' what one expects.  OpenSchematic is a two field array of elements (0)
 ' and (1).
    MsgBox "OpenSchematicCount="& OpenSchematicCount & vbCrLf &_
        "OpenSchematic (0)=" & OpenSchematic(0) & vbCrLf &_
        "OpenSchematic (1)=" &_
       OpenSchematic(1),,"OpenSchematic Array"
    ' Use the Documents Open method to open the desired schematic.
     Set vddoc = vdapp.SchematicSheetDocuments.Open
 (OpenSchematic(0),OpenSchematic(1))
     vddocName = vddoc.Name
     MsgBox "Vddoc Name is: " & vddocName
     ' Create the document and View objects.
     Set vddoc = vdapp.ActiveDocument
     Set vdview = vdapp.ActiveView
     'Set the NoAppOpen variable False to terminate While loop.
     NoAppOpen = False
    Else
  '    If conditional not satisfied, therefore the user needs to see
     'the prompt again.
     NoAppOpen = True
       End If
      Wend
 Else
  ' Application is running message to the user
  MsgBox "Application is running",,"Empty Check"
  ' Use the GetObject method to acquire a "link" to the application
     Set vdapp = GetObject(,"ViewDraw.Application")
     ' Create the document object
     Set vddoc = vdapp.ActiveDocument
     ' Test to see if a drawing is open or only the application
     If vddoc Is Nothing  Then
       MsgBox "No Schematic is open.",,"Schematic Test"
' Prompt the user for a schematic to open if there isn't an open document.
While NoDocOpen
If Not OpenMySchematic="NameRequired" Or Not OpenMySchematic="" Then
```

```
    ' Prompt user for the desired schematic name and page number.
OpenMySchematic = InputBox ("Schematic Name, Page Number (e.g. Top,1): ",_
            "User Prompt","NameRequired",_
              500, 500)
 ' Split the user input so the user input can be put in the appropriate
 ' format for the SchematicSheetDocuments.Open method below.
      OpenSchematic = Split (OpenMySchematic,",",-1)
   ' Use the UBound function to get the upper limit of the resulting
   ' array.  This could also be used with some code as an "error"
' check (e.g. did the user enter the right number of strings?, etc.)
      UboundOpenSch = UBound (OpenSchematic)
 ' The array generated by the Split function is a zero reference array.
 ' So, the actual quantity of elements is the UBound value plus 1.
      OpenSchematicCount = UboundOpenSch + 1
 ' A message box is a good way of displaying the data to ensure that it
 ' what one expects.  OpenSchematic is a two field array of elements (0)
 ' and (1).
    MsgBox "OpenSchematicCount="& OpenSchematicCount & vbCrLf &_
   "OpenSchematic (0)=" & OpenSchematic(0) & vbCrLf &_
   "OpenSchematic (1)=" & OpenSchematic(1),,"OpenSchematic Array"
 ' Use the Documents Open method to open the desired schematic.
Set vddoc = vdapp.SchematicSheetDocuments.Open_
(OpenSchematic(0),OpenSchematic(1))
     ' Create the Document and View objects
     Set vddoc = vdapp.ActiveDocument
     Set vdview = vdapp.ActiveView
     NoDocOpen = False
    Else
     NoDocOpen = True
       End If ' OpenMySchematic
      Wend  ' NoDocOpen
      ' Create the View object if the document exists.
   Set vdview = vdapp.ActiveView
     End If ' IsEmpty vddoc
     Set vdview = vdapp.ActiveView
 End If ' IsEmpty Application
 On Error GoTo 0
' Set a string variable to contain the derived project directory.
 CurProjDir = vdapp.GetProjectData.GetProjectPath
 MsgBox "CurProjDir is " & CurProjDir
' Create an object variable, CurBlock, whose value is the current block.
' Will be used for a statement which deselects everything on the page.
Set CurBlock = vdview.Block
' Clears the status bar.
vdapp.StatusBarText("")

' Create a filesystem object and text file
Set FSO = CreateObject("Scripting.FileSystemObject")
Set CompRefXref = FSO.CreateTextFile(CurProjDir & "\CompRefXref.txt", True)

' Write a header line to the text file.
CompRefXref.Write "Inst REFDES" & vbTab & "Symbol REFDES" & vbTab & "Xloc"
 & vbTab & "Yloc" & vbTab & "Internal ID " & vbCrLf
```

```
' Create an Excel file with the same data as in the .txt file
Dim xl, ActSheet, ActCell, wb
Dim inc
inc = 2
 Set xl = CreateObject("Excel.Application")
 xl.Visible = True
 Set wb = xl.Workbooks.Add
 Set ActSheet = wb.ActiveSheet
 Actsheet.name="RefDes Crossreference"
 Set ActCell = xl.Cells(1,1)
 ActCell.Value="Inst REFDES"
 Set ActCell = xl.Cells(1,2)
 ActCell.Value="Symbol REFDES "
 Set ActCell = xl.Cells(1,3)
 ActCell.Value="Xloc"
 Set ActCell = xl.Cells(1,4)
 ActCell.Value="Yloc"
 Set ActCell = xl.Cells(1,5)
 ActCell.Value="Internal ID"

' Create a collection of components from the Xpedition Designer schematic.
Set CompColl = vdview.query(VDM_COMP, VD_ALL)
' Populate the CompCollCount Variable with number of components to process
CompCollCount = CompColl.count

' Process each element of the collection.
For Each CompObj In CompColl
 ProcCompCount = ProcCompCount + 1
' Issue a message to the status bar.
 vdapp.StatusBarText("Processing component " & ProcCompCount & " of " &
 CompCollCount & " components.")

 ' Identify the symbol block type.  Required for hierarchical traversal
 ' although, its purpose in this script is to determine if it we should
 ' check for a REFDES.
 CompSymType = CompObj.SymbolBlock.SymbolType
 ' Select the component
 CompObj.Selected = True
 ' Zoom in on the selected component
 vdapp.executecommand "zselect"

 ' Test for block type of Pin, then get attributes.
 ' Could be used to identify the NETNAME attribute value which
 ' becomes the implicit netname of the net connected to the
 ' component object whose symbol type is Pin.
 If CompSymType = 4 Then
  MsgBox "Block type Pin; No REFDES.",,"Block Type Test"
 ElseIf CompSymType = 3 Then
  MsgBox "Block type Annotate; No REFDES.",,"Block Type Test"
 Else
  ' Create a variable whose value represents the REFDES attribute
  ' object within the schematic.
  Set RefDesAttr = CompObj.FindAttribute("REF DESIGNATOR")
```

```
    ' Test for presence of REFDES
    If RefDesAttr Is Nothing Then
     MsgBox "Part will not be processed. No RefDes found.",,"Refdes?"
    Else

   ' Create two variables whose values hold the name and value of the
   ' Refdes attribute.
     RefDesName = RefDesAttr.Name
     RefDesVal  = RefDesAttr.value
     RefDesInstValue = RefDesAttr.InstanceValue
' Instance level value of attribute

  ' Create two variables whose values hold the x and y coordinates of the
  ' component.
     CompXLoc = CompObj.GetLocation.X
     CompYLoc = CompObj.GetLocation.Y

' Write to a text file.  The vbTab and the "white space" are used
' formatting technique examples.
CompRefXref.WriteLine "    " & RefDesInstValue & vbTab & vbTab & "    " &_
RefDesVal  & "     " & vbTab &  CompXLoc & vbTab & CompYLoc & vbTab &
 CompObj.UID

     ' Write to the Excel file
      Set ActCell = xl.Cells(inc,1)
      ActCell.Value = RefDesInstValue
      Set ActCell = xl.Cells(inc,2)
      ActCell.Value = RefDesVal
      Set ActCell = xl.Cells(inc,3)
      ActCell.Value = CompXLoc
      Set ActCell = xl.Cells(inc,4)
      ActCell.Value = CompYLoc
      Set ActCell = xl.Cells(inc,5)
      ActCell.Value = CompObj.UID

     ' Increment the row count
         inc = inc +1

    End If  ' RefDesAttr
  End If  ' CompSymType
'  Deselect the parts
    CurBlock.DeselectAll
  Next
' Close the text file.
CompRefXref.Close
'Set View of schematic to full page
vdview.ViewFull
' Issue a message to the status bar.
 vdapp.StatusBarText("Script successfully completed! Excel has launched
 with the results.")

' Launch Excel on text file created by the FSO.CreateTextFile method.
```

```
Dim Win
Set win =CreateObject("WScript.shell") ' Create the windows object
'Use the run method to launch an application
win.run "notepad.exe " & CurProjDir & "\CompRefXref.txt"
```

Example 4: Using Applications and Documents in Scripts

This script demonstrates the usage of some of the Application level methods/properties/events.

The script demonstrates these concepts best when multiple sheets and multiple schematics are opened before running the script. The script expects a application window to be open and one, or more schematics to be open.

The queries at the end of the script demonstrate the scope of the query depending on the object used.

The object variables have the following usage scopes:

- **vdapp** — Xpedition Designer application
- **vddoc** — Xpedition Designer document (the active document)
- **vddocColl** — Xpedition Designer documents (open files in Xpedition Designer)
- **vdview** — Current view

Note:
Keep in mind that the variable values must be refreshed when navigating the design to ensure that any scripting object is reporting/operating on the desired design data.

You can use this script via any of the methods described in "Running Scripts" on page 25.

Note:
Mentor Graphics recommends that you use COM versioning syntax in script examples that use GetObject and CreateObject. Without COM versioning, the script will access the last installation to which the release switcher pointed.

```
Option Explicit

Dim VDM_COMP, VD_ALL, VDFILL_VERT, VD_SELECTED
Dim vdapp,vddoc, vdview, vddocColl, Comp, CompRefdes, inc
Dim ProjDir,ProjName
Dim Index, ItemObj, ItemName

'
' The following constants are known within Xpedition Designer and
' DO NOT need to be defined when the script is run from within
' Xpedition Designer.  Running the script as a client using a debugger,
' Xpedition Designer doesn't understand it's internal enumerated variables
```

```
' and their values, so we are providing them here.
'
' Once a script has been debugged, the intention is to run them
' within the Xpedition Designer application, it is not necessary to set
' the enumerations as they will be provided within the application.
'
 VDM_COMP =  128
 VD_ALL = 0
 VD_SELECTED = 1
 VDFILL_VERT = 13
 inc = 1

' The following 2 lines are allow for the Xpedition Designer Application
' to be invoked and adds the type library.
' When using an external debugger, you may need these lines,
' even though they are not used, in order for the debugger to load the
' application's Type library.
'
 'Set vdapp = CreateObject("ViewDraw.Application")
 ' invokes viewdraw.exe and gives you back the app object
 'vdapp.Scripting.AddTypeLibrary("ViewDraw.Application")
 ' Adds the Xpedition Designer Type library.

' Attach to a running Xpedition Designer process

 Set vdapp = GetObject(,"ViewDraw.Application")
 vdapp.visible = True

' Create a view object which points to the current active
' drawing displayed in Xpedition Designer

 Set vdview = vdapp.ActiveView

' Create a Document object which points to the current active
' document in memory for Xpedition Designer
 Set vddoc  = vdapp.ActiveDocument
 MsgBox "ActiveDocument Name: " & vddoc.name,,"Vddoc Name"

'
'  ****** Begin Flat design navigation example  *******
'
 Dim CurBlock, CurBlockNamePage
' Create the block object
 Set CurBlock = vdview.block

' Create a variable whose value is the filename and extension.
 CurBlockNamePage = CurBlock.GetName(1) & "." & CurBlock.SheetNum
 MsgBox "The current schematic and page is : " & CurBlockNamePage,,"Initial
 Block/Page"

' Push to the next page using the Xpedition Designer Command line command
 vdapp.ExecuteCommand "PSH" 'Pushes to next page.
 Set vdview = vdapp.ActiveView
 Set CurBlock = vdview.block
```

```
 CurBlockNamePage = CurBlock.GetName(1) & "." & CurBlock.SheetNum
 MsgBox "The current schematic and page is : " &
 CurBlockNamePage,,"Block/Page Post Push"

' Return to the previous view using the Xpedition Designer Pop command
' line command
 vdapp.ExecuteCommand "Pop"' Create the block object
 Set vdview = vdapp.ActiveView
 Set CurBlock = vdview.block
 CurBlockNamePage = CurBlock.GetName(1) & "." & CurBlock.SheetNum
 MsgBox "The current schematic and page is : " &
 CurBlockNamePage,,"Block/Page Post Pop"

'
' ******    End Flat design navigation example  *******
'
'
' ******    Begin Adding box and setting object property example ******

 MsgBox "About to add a box",,"Advise Adding Box"

' Use the AddBox method to add a box. Requires two xy coordinates
 CurBlock.AddBox 50,50,350,350 ' Form:
 x(lowerleft),y(lowerleft),x(upperright),y(upperright),
' The box which was just added is still selected.
 Dim gBoxColl,gBox, VDM_BOX: VDM_BOX = 2
'
' The following statements create a collection of the selected boxes,
' then, for each selected box found, sets the fill style to vertical,
' deletes the box, then performs an undo, then deselects the box using
' the box object Selected property.
'
 Set gBoxColl = vdview.query (VDM_BOX, VD_Selected)
  For Each gBox In gBoxColl
   ' Set fill style to vertical
   gBox.fillstyle(VDFILL_VERT)
   ' Refresh the graphics so the vertical cross hatch is displayed.
   vdapp.executecommand "refresh"
   MsgBox "Filled the box",,"What did I do to the box?"
   MsgBox "About to Delete Selected",,"Deleting.."
   ' Demonstrating the DeleteSelected method
   ' (parameter)False leaves dangling nets; True removes them.
   ' As there aren't any nets on the graphical box, either seting
   ' would be ok.  Must be correctly specified for components.
   CurBlock.DeleteSelected(False)
   MsgBox "Deleted the box",,"Advisory"
   ' Executing a command line command to undo the deletion.
   vdapp.ExecuteCommand "undo"
   ' Deselecting the box. Same as clicking the Left mouse button
   ' in an unoccupied are of the drawing.
   gBox.Selected = False
  Next
 MsgBox "Box Added, Check schematic",,"Add Box Verify"
```

```
'
' *******   End Adding box and setting object property example ******
'

'
' *******  Begin application visibility control example.
'

' The following lines toggle the visibility of the Application
 (Xpedition Designer window).  The message boxes simply allow the user to
 see the
' application visibility set to invisible.

 vdapp.Visible = False
 MsgBox "Just made the Application invisible",,"What Happened?"
 vdapp.visible =True
 MsgBox "Just made the Application visible",,"What Happened Now?"

'
' *******  End application visibility control example.
'
' *******  Begin Project and software information extraction example ****
'
 'Assigns the primary directory pointer to the variable ProjDir
 'and the project Name to the variable ProjName, then displays
'   both in a message box

 ProjDir = vdapp.GetProjectData.GetProjectPath
 ProjName = vdapp.GetProjectData.GetProjectName
 MsgBox "Project Name is " & ProjName & vbCrLf &_
     "Project Directory is: " & ProjDir,,"Project information"

 MsgBox vdapp.Name & "  " & vdapp.version & vbCrLf &_
     "Executable " & vdapp.fullname,,"Xpedition Designer Information"


' ViewFull method resets CurrentView to full screen.
 Dim CurrentView
 Set CurrentView = vdapp.ActiveView
 CurrentView.viewfull
'
' *******  End Project and software information extraction example ******
'


'
' ******* Begin Status Bar control example
'
 vdapp.StatusBarText("Starting Test")
'
' ******* End Status Bar control example
 '

 MsgBox "Setting Xpedition Designer window non-interactive.",,"Disable
 mouse/keyboard Input"
```

```
 vdapp.interactive = false ' Xpedition Designer Application window will
        ' not respond to mouse/keyboard entries
        ' until vdapp.interactive is set to
        ' True.  This includes all window controls
        ' such as minmize/maximize, etc.

 vdapp.busycursor = True  ' Set the cursor to the hour glass.

 MsgBox "Xpedition Designer: Mouse/keyboard input disabled",,"Disable
Message"

 ' Issue a message to the status bar.
 vdapp.StatusBarText("Script has disabled this Xpedition Designer Window
 from accepting any user input")

'
' *******  End Disabling the Application window example   ********
 '

'  ***** Begin cycling through and reporting on all the components on
'        the open documents
'
'  Cycle through the open documents and display component information.
' Uses a copy of some code from above to activate the different documents.
'
 ' Create a collection of documents
 Set vddocColl = vdapp.documents
 ' Process all documents using index to "target" each in the collection.
 For Index=1 To vddocColl.Count  ' Number of open documents
  Set ItemObj = vddocColl.Item(Index)
  ' Create a document object for members of the collection
  ItemName = ItemObj.Name ' Document object's "Name" property
     MsgBox "Document Name: " & ItemName & vbCrLf &_
         "Document Index: " & Index,,"Document Information"
'
' Document Activate example
'
  MsgBox "Changing the active document to " & ItemObj.Name,,"Changing
 Selected Tab"
  ItemObj.activate()

  Set vdview = vdapp.ActiveView
  ' Create the block object
  Set CurBlock = vdview.block
   For Each Comp in vdview.Query(VDM_COMP, VD_ALL)
  ' Queries ONLY the current active
  ' schematic/symbol file in Xpedition Designer
  ' (e.g. queries only the current "view")
    ' Select the component
    Comp.selected = True
    ' Zoom in on each component as it's being processed.
    vdapp.executecommand "zselect"
    ' Set variable CompRefdes to the Refdes value
```

```
    Set CompRefdes = Comp.FindAttribute("Ref Designator")

    ' Test for the existence of the REFDES attribute
     If CompRefdes Is Nothing Then
         MsgBox  "No REFDES attribute found and component count is "_
        & inc & vbCrLf &_
           "Component ID " & Comp.uid & vbCrLf &_
           "Current document is " &_
        vdapp.activeDocument.name,,_
        "Component Info - No Refdes"
     Else
      MsgBox "REFDES: " & CompRefdes.Value &_
        " and component count is " & inc & vbCrLf &_
          "Component ID " & Comp.uid & vbCrLf &_
          "Current document is " &_
        vdapp.activeDocument.name,,_
        "Component Info for Comps with REFDES"
     End If
    inc = inc + 1
    ' Deselect comp and its attributes. Setting the Comp.selected
    ' property to False will not deselect the its attributes.
    CurBlock.deselectall
   Next ' Each Component
 Next ' Each Document

 MsgBox "End of Script",,"Script Ending Dialog"

'
'  *******   Begin Enabling the Application window example *******
'
 ' Change the cursor back to the pointer
 vdapp.busycursor = False   ' True sets cursor graphic to hour glass;
          ' false sets graphic to pointer
          ' The pointer itself, however remains active.
 ' Enable the application window to respond to mouse and keyboard input.
 vdapp.interactive = True   ' True enables GUI interaction; False
          ' disables mouse and keyboard inputs

'
'  *******   End Enabling the Application window example *******
'


 ' Issue a message to the status bar.
 ' Displays the string in Xpedition Designer status bar in the lower
   ' left of the window frame.
 vdapp.StatusBarText("Script finished successfully!")
```

Chapter 3
Schematic Editor Data Objects

This section contains the alphabetical listing of Xpedition Designer Automation Objects.

The following table includes summary information for each object you can access in Xpedition Designer Automation. To view the full description for a specific object, click the Object name.

Table 3. Xpedition Designer Objects

Object	Description
AddinInfo Object	The AddinInfo object defines the characteristics necessary to load an addin into the application.
Application Object	The Application object represents Xpedition Designer application itself. Generally, it is the starting point for working with the automation capabilities of the application.
Arc Object	This object represents the Xpedition Designer arc graphical object.
Attribute Object	This object represents an attribute of an object (a net, component, block, or other object) on a schematic.
Block Object	This object represents a single sheet of a schematic or symbol.
Box Object	The Box is a graphical object in Xpedition Designer.
CColor Object	The CColor object represents a color in RGB format.
Circle Object	This object represents a graphical circle on a schematic.
CommandsManager Object	The CommandsManager object allows the user to perform command-related actions: (un)register, execute, enable/disable commands).
Component Object	This object represents a component placed on a schematic.
ComponentPin Object	This object represents a pin associated with a component on a schematic.
Connection Object	A connection object represents a component-to-net, net-to-bus, or bus-to-bus connection on a schematic.
DesignerDataManagement Object	You can access EDM managed designs using the DesignerDataManagement object. The following table lists methods and properties of the DesignerDataManagement object with links to the respective reference pages.
DesignerDataManagementAdditionalProperty Object	You can access EDM managed designs using the DesignerDataManagementAdditionalProperty object. The following table lists methods and properties of the DesignerDataManagementAdditionalProperty object with links to the respective reference pages.
DesignerDataManagementEntity Object	You can access EDM managed designs using the DesignerDataManagementEntity object. The following table lists methods and properties of the DesignerDataManagementEntity object with links to the respective reference pages.
HDLSourceDocument Object	This object represents an HDL source file document that has been opened in Xpedition Designer.
Label Object	The label object represents the label of an object in Xpedition Designer.
Line Object	The Line Object represents a graphical line in a schematic.

Table 3. Xpedition Designer Objects (continued)

Object	Description
Net Object	This object represents a net or bus on a schematic.
PDBPartitions Object	This object allows the user to manipulate PDB partitions data in a Xpedition Designer project file.
Pin Object	This object represents a symbol pin on a schematic.
Point Object	This object represents a point on a schematic specified by its X,Y coordinates.
ProjectData Object	ProjectData is an automation object which allows managing data stored in Xpedition Designer project file and provides some basic file-related information.
Rect Object	This object represents a graphic rectangle on a schematic, specified by its coordinates.
Ripper Object	This object represents a bus ripper on a schematic.
SchematicSheetDocument Object	This object represents one sheet of a schematic.
Segment Object	This object represents a portion of a net or bus on a schematic.
ShellExecution Object	The ShellExecution object defines the characteristics necessary to run an external program in the application.
SymbolPartitions Object	This object allows you to manipulate symbol partitions data in a project file.
Text Object	This object represents text on a Schematic.
View Object	This object encapsulates the graphics display of a document (either a schematic or a symbol).
Viewport Object	The Viewport object encapsulates the Xpedition Designer graphics interface. Using the methods and properties associated with this object allows you to draw graphics onto a View.

AddinInfo Object

The AddinInfo object defines the characteristics necessary to load an addin into the application.

The following tables list methods, properties and events of the AddinInfo object with links to the respective reference pages:

Table 4. AddinInfo Object Properties

Property	Description
InitiallyDisabled Property (AddinInfo Object)	Specifies whether or not the addin is disabled when the host application is invoked.
InitiallyVisible Property (AddinInfo Object)	Specifies whether or not the addin is visible when the host application is invoked.
LicenseFeature Property (AddinInfo Object)	Gets or sets the license feature of the addin.
Name Property (AddinInfo Object)	Returns or sets the name of the addin.
Placement Property (AddinInfo Object)	Returns or sets the placement of the addin.
ProgId Property (AddinInfo Object)	Returns or sets the program identification of the addin.
RuntimeCreateDecision Property (AddinInfo Object)	Indicates whether or not the creation decision must be delegated to the module that hosts the addin.
ShortCutKey Property (AddinInfo Object)	Returns or sets the shortcut key associated with the addin.
ToolbarButton Property (AddinInfo Object)	Indicates whether or not the addin must have a toolbar button.

InitiallyDisabled Property (AddinInfo Object)

Scope: Schematic editor

Object: AddinInfo Object

Access: Read/Write

Prerequisites: None

Specifies whether or not the addin is disabled when the host application is invoked.

Usage

AddinInfo.**InitiallyDisabled**

Arguments

None

Return Values

Boolean. A value of “True” indicates that the addin is disabled upon invocation of the host application; “False” indicates that the addin is enabled.

InitiallyVisible Property (AddinInfo Object)

Scope: Schematic editor

Object: AddinInfo Object

Access: Read/Write

Prerequisites: None

Specifies whether or not the addin is visible when the host application is invoked.

Usage

AddinInfo.**InitiallyVisible**

Arguments

None

Return Values

Boolean. A value of “True” indicates that the addin is visible upon invocation of the host application; “False” indicates that the addin is invisible.

LicenseFeature Property (AddinInfo Object)

Scope: Schematic editor

Object: AddinInfo Object

Access: Read/Write

Prerequisites: None

Gets or sets the license feature of the addin.

Usage

AddinInfo.**LicenseFeature**

Arguments

None

Return Values

String. Specifies the value of the license feature for the addin.

Name Property (AddinInfo Object)

Scope: Schematic editor

Object: AddinInfo Object

Access: Read/Write

Prerequisites: None

Returns or sets the name of the addin.

Usage

AddinInfo.**Name**

Arguments

None

Return Values

String. The string that contains the name of this addin.

Placement Property (AddinInfo Object)

Scope: Schematic editor

Object: AddinInfo Object

Access: Read/Write

Prerequisites: None

Returns or sets the placement of the addin.

Usage

AddinInfo.**Placement**

Arguments

None

Return Values

String. The string that specifies the placement for the addin.

Description

This string may have one of the following values:

- Right
- Bottom
- Left

ProgId Property (AddinInfo Object)

Scope: Schematic editor

Object: AddinInfo Object

Access: Read/Write

Prerequisites: None

Returns or sets the program identification of the addin.

Usage

AddinInfo.**ProgId**

Arguments

None

Return Values

String. The string that contains the program identification of this addin.

RuntimeCreateDecision Property (AddinInfo Object)

Scope: Schematic editor

Object: AddinInfo Object

Access: Read/Write

Prerequisites: None

Indicates whether or not the creation decision must be delegated to the module that hosts the addin.

Usage

AddinInfo.**RuntimeCreateDecision**

Arguments

None

Return Values

Boolean. True indicates that the creation decision must be delegate to the host module; False otherwise.

ShortCutKey Property (AddinInfo Object)

Scope: Schematic editor

Object: AddinInfo Object

Access: Read/Write

Prerequisites: None

Returns or sets the shortcut key associated with the addin.

Usage

AddinInfo.**ShortCutKey**

Arguments

None

Return Values

String. The string that contains the shortcut key notation for this addin.

Key combinations are concatenated with a “+” symbol. For example, “Ctrl+Alt+J”.

ToolbarButton Property (AddinInfo Object)

Scope: Schematic editor

Object: AddinInfo Object

Access: Read/Write

Prerequisites: None

Indicates whether or not the addin must have a toolbar button.

Usage

AddinInfo.**ToolbarButton**

Arguments

None

Return Values

Long. “1” indicates that the addin must have a toolbar button; “0” otherwise.

Application Object

The Application object represents Xpedition Designer application itself. Generally, it is the starting point for working with the automation capabilities of the application.

You can get access to the existing instance of application or create it (launch application).

The following tables list methods, properties and events of the Application object with links to the respective reference pages:

Table 5. Application Object Methods, Properties, and Events

Method, Property, or Event	Description
Activate Method (Application Object)	Makes the application window the active window.
AddAddin Method (Application Object)	Adds an addin to the application.
AppendOutput Method (Application Object)	Adds text to the specified tab of the Output window. If the tab does not already exist, the application creates it.
CloseProject Method (Application Object)	Closes the application project.
CommandsManager Method (Application Object)	Allows access to the CommandsManager object.
DesignComponents Method (Application Object)	Builds a collection of component objects for a schematic or an entire design.
DesignNets Method (Application Object)	Builds a collection of net objects for a schematic or an entire design.
DesignPaths Method (Application Object)	Builds a collection of component paths for the entire design.
GetActiveDesign Method (Application Object)	Retrieves the name of the active design.
GetDefaultColor Method (Application Object)	Returns the color (in RGB format) for the specified object type.
GetProjectData Method (Application Object)	Returns the ProjectData object.
Initialize Method (Application Object)	Initalizes the application.
MGCDesignerDataManagement Method (Application Object)	Returns the designer data management object.
NewProject Method (Application Object)	Creates a new Xpedition Designer project. The new project folder must not already exist.
OpenBlocks Method (Application Object)	Returns a collection of Block objects.
OpenProject Method (Application Object)	Opens a project.
OpenURL Method (Application Object)	Opens a document at the location specified by a URL in the application.
ParamGetMode Method (Application Object)	Returns the global mode setting that controls how Xpedition Designer operates.

Table 5. Application Object Methods, Properties, and Events (continued)

Method, Property, or Event	Description
ParamGetValue Method (Application Object)	Returns the global value setting that controls how Xpedition Designer operates.
ParamSetMode Method (Application Object)	Sets the global mode setting that controls how Xpedition Designer operates.
ParamSetValue Method (Application Object)	Sets the global value setting that controls how Xpedition Designer operates.
PrintProject Method (Application Object)	Prints an entire design, or a portion of a design from the top level down to the level specified.
ProductionLibraryLimitationsChanged Method (Application Object)	Updates the Variant Manager Settings dialog box when the production library limitation is modified.
PushPath Method (Application Object)	Opens a schematic sheet by pushing through the components of a path.
Query Method (Application Object)	Returns a collection of objects based on type.
QueryPages Method (Application Object)	Returns a collection of block objects.
Quit Method (Application Object)	Terminates the application.
RunISE Method (Application Object)	Opens an existing non-local symbol in the embedded symbol editor. Otherwise, creates a new empty symbol with a given name.
SchematicSheetDocuments Method (Application Object)	Returns a collection of SchematicSheetDocument objects.
SelectPath Method (Application Object)	Opens a schematic and selects objects based on a path from the top of the design.
SelectPathCompPin Method (Application Object)	Opens a schematic and selects a component pin based on a path from the top of the design.
SetDefaultColor Method (Application Object)	Sets the new color (in RGB format) for specified object type.
SetRedraw Method (Application Object)	Allows or prevents changes from being drawn.
StartMigration Method (Application Object)	Allows you to perform a project migration from EXP2005.1 to EXP 2007.
ActiveDocument Property (Application Object)	Returns the ActiveDocument collection.
ActiveView Property (Application Object)	Returns the ActiveView collection for the currently activated schematic or symbol.
Addins Property (Application Object)	Allows access to an Addins collection.
CommandBars Property (Application Object)	Returns a collection of CommandBar objects.
CommandLineArguments Property (Application Object)	Returns the command line arguments that were used to call the application.
Interactive Property (Application Object)	Controls the response of the application to activity on its user interface.
QueueSelectEvents Property (Application Object)	Sets or disables event queuing.

Table 5. Application Object Methods, Properties, and Events (continued)

Method, Property, or Event	Description
ShellCmd Property (Application Object)	Returns the ShellCmd collection, which allows you to execute programs.
SilentMode Property (Application Object)	Returns or sets the silent mode status of the application.
SourceDocuments Property (Application Object)	Returns a collection of HDLSourceDocument objects.
StatusBarText Property (Application Object)	Returns or sets the text that is displayed in the application status bar.
Version Property (Application Object)	Returns the version number of the application that appears when you choose About Xpedition Designer on the Help menu.
Visible Property (Application Object)	Returns or sets the visibility of the application.
ActivateView Event (Application Object)	Occurs when the View child window becomes active.
ActivateView2 Event (Application Object)	Occurs when the View child window becomes active (receives focus), not when the application becomes active.
AfterDocumentOpened Event (Application Object)	Occurs after a document is opened and drawn.
AfterPrintProject Event (Application Object)	Occurs after a print project is completed.
AfterSheetRead Event (Application Object)	Occurs after a sheet is read into memory.
AfterSheetReRead Event (Application Object)	Occurs after a sheet has been reread.
BeforeDocumentOpened Event (Application Object)	Occurs before a document is opened and drawn.
BeforePrintProject Event (Application Object)	Occurs before a print project is executed.
BeforeProjectChanged Event (Application Object)	Occurs before the project is changed. That is, the event occurs before a opening another project.
BlockLocked Event (Application Object)	Occurs when you change the state of a schematic block from read-only to read/write, using the “Click to Edit” button.
BlockModified Event (Application Object)	Occurs every time a Block Object is modified.
CreateObject Event (Application Object)	Occurs whenever a new object is created. Use this event to stipulate other events, such as verifying that a component is labeled, has correct attributes, and so on.
DeactivateView Event (Application Object)	Occurs when a child window is deactivated. This event is not triggered when the Application is deactivated.
DeactivateView2 Event (Application Object)	Occurs when a child window is deactivated. This event is not triggered by deactivating the Application.
Delete Event (Application Object)	Occurs whenever an object is deleted.
DocumentClose Event (Application Object)	Occurs whenever a SchematicSheetDocument Object is closed.
LockRequest Event (Application Object)	Occurs once before modifying a Block Object.

Table 5. Application Object Methods, Properties, and Events (continued)

Method, Property, or Event	Description
MouseMoved Event (Application Object)	This event allows you to track mouse movements and occurs when the mouse is moved in a View.
PaintRegion Event (Application Object)	Occurs when a View is painted.
PrintFile Event (Application Object)	Occurs when a file is printed.
ProductionLibraryLimitationsChanged Event (Application Object)	Occurs when the production library limitation is changed.
ProjectChanged Event (Application Object)	Occurs after a project is changed and then re-opened.
ProjectClosed Event (Application Object)	Occurs after a project is closed.
Select Event (Application Object)	Occurs when an object is selected or unselected. You can use this event to access such things as crossprobing.
Shutdown Event (Application Object)	Occurs when Xpedition Designer shuts down.
SourceDocumentSave Event (Application Object)	Occurs when a source document (VHDL, Verilog, or Spice) is saved.
SourceFileModified Event (Application Object)	Occurs when an existing source document (VHDL, Verilog, or Spice) is modified and saved.
Startup Event (Application Object)	Occurs when Xpedition Designer starts.
SymbolPreviewed Event (Application Object)	Occurs when the symbol previewer displays a new symbol.
Unlock Event (Application Object)	Occurs when a block is being unlocked (the application saves a block or closes a block without saving it).

Activate Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Makes the application window the active window.

Usage

Application.**Activate**()

Arguments

None

Description

If the window is obscured, this method pops it to the front of the screen. Typically, automation scripts will include this method in order to activate the schematic editor session before performing other tasks.

AddAddin Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Adds an addin to the application.

Usage

Application.**AddAddin**(ByVal *pAddinInfo* As IAddinInfo) As Boolean

Arguments

- pAddinInfo

  The definition of the addin that is added to the application.

Return Values

As Boolean. Returns True if the addin was successfully added; False otherwise.

AppendOutput Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Adds text to the specified tab of the Output window. If the tab does not already exist, the application creates it.

Usage

Application.**AppendOutput**(ByVal *TabName* As String, ByVal *String* As String)

Arguments

- TabName

  A string representing the name of the tab that is to be added to the Output window.
- String

  A string representing the text to be added to the Output window.

Examples

This example appends text to the Output tab, and creates a new tab “My Tab” to which it also appends text.

Note:
Mentor Graphics recommends that you use COM versioning syntax in script examples that use GetObject and CreateObject. Without COM versioning, the script will access the last installation to which the release switcher pointed.

```
Option Explicit
Dim vdapp
Set vdapp = GetObject (,"ViewDraw.Application")
Call vdapp.AppendOutput("Output", "Appending Text to ""Output"" tab")
Call vdapp.AppendOutput("My Tab", "Appending Text to ""My Tab"" tab")
```

Appending Text to "My Tab" tab
Output
Output
My Tab
For Help, press F1

CloseProject Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Closes the application project.

Usage

Application.**CloseProject**() As Boolean

Arguments

None

Return Values

As Boolean. Returns True if the project was closed successfully. Returns False otherwise.

CommandsManager Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Allows access to the CommandsManager object.

Usage

Application.**CommandsManager**() As ICommandsManager

Arguments

None

Description

CommandsManager is an automation object that allows user to perform command-related actions: (un)register, execute, enable/disable commands.

DesignComponents Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Builds a collection of component objects for a schematic or an entire design.

Usage

Application.**DesignComponents**(ByVal *Library* As String, ByVal *Name* As String, [ByVal *EndSheet* As String], [ByVal *LevelString* As String], [ByVal *RecurseDown* As Boolean = True]) As IVdObjs

Arguments

- Library

  Specific library to search. This argument may be empty. When empty, the schematic editor will look through the entire search order.

- Name

  Name of the design that contains the components to be collected.

- EndSheet

  (Optional) Specifies the name of the last sheet for which objects are to be collected. A value of “-1” (default) indicates that all sheets be included.

- LevelString

  (Optional) This argument is a space delimited string that specifies the levels at which to stop loading anything other than primitives. For example, STD VHDL would stop at the primitive, on the STD and VHDL levels of the hierarchy. This argument is ignored if RecurseDown is False.

- RecurseDown

  (Optional) Indicates whether or not to traverse down the hierarchy. If RecurseDown is True then the Levels string is tested against each component instance before traversing down. If the flag is False then only sheets at the same level of hierarchy are traversed.

  True (default) - Traverse into the hierarchy. False - Do not traverse into the hierarchy.

Return Values

As IVdObjs. The collection of Component Objects.

Description

You can get all components in the hierarchy or simply all components at the same level of the hierarchy by setting the RecurseDown flag. If there are multiple paths to a schematic, the components on the schematic will be in the collection multiple times.

DesignNets Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Builds a collection of net objects for a schematic or an entire design.

Usage

Application.**DesignNets**(ByVal *Library* As String, ByVal *Name* As String, [ByVal *EndSheet* As String], [ByVal *LevelString* As String], [ByVal *RecurseDown* As Boolean = True]) As IVdObjs

Arguments

- Library

  Specific library to search. This argument may be empty. When empty, the schematic editor will look through the entire search order.

- Name

  Name of the design that contains the nets to be collected.

- EndSheet

  (Optional) Specifies the name of the last sheet for which objects are to be collected. A value of “-1” (default) indicates that all sheets be included.

- LevelString

  (Optional) This argument is a space delimited string that specifies the levels at which to stop loading anything other than primitives. For example, STD VHDL would stop at the primitive, on the STD and VHDL levels of the hierarchy. This argument is ignored if RecurseDown is False.

- RecurseDown

  (Optional) Indicates whether or not to traverse down the hierarchy. If RecurseDown is True then the Levels string is tested against each component instance before traversing down. If the flag is False then only sheets at the same level of hierarchy are traversed.

  True (default) - Traverse into the hierarchy. False - Do not traverse into the hierarcy.

Return Values

As IVdObjs. The collection of Net Objects.

Description

You can get all nets in the hierarchy or simply all nets at the same level of the hierarchy by setting the RecurseDown flag. If there are multiple paths to a schematic, the nets on the schematic are in the collection multiple times.

DesignPaths Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Builds a collection of component paths for the entire design.

Usage

Application.**DesignPaths**(ByVal *Library* As String, ByVal *Name* As String, ByVal *LevelString* As String, ByVal *RecurseDown* As Boolean) As IStringCollection

Arguments

- Library

  Specific library to search. This argument may be empty. When empty, the schematic editor will look through the entire search order.

- Name

  The name of the target design.

- LevelString

  (Optional) This argument is a space delimited string that specifies the levels at which to stop loading anything other than primitives. For example, STD VHDL would stop at the primitive, on the STD and VHDL levels of the hierarchy. This argument is ignored if RecurseDown is False.

- RecurseDown

  (Optional) Indicates whether or not to traverse down the hierarchy. If RecurseDown is True then the Levels string is tested against each component instance before traversing down. If the flag is False then only sheets at the same level of hierarchy are traversed.

  True (default) - Traverse into the hierarchy. False - Do not traverse into the hierarchy.

Return Values

As IStringCollection. This is the StringCollection Collection of component paths for the design.

GetActiveDesign Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Retrieves the name of the active design.

Usage

Application.**GetActiveDesign**() As String

Arguments

None

Return Values

As String. The name of the active design. If there is currently no active design, an empty string is returned.

GetDefaultColor Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Returns the color (in RGB format) for the specified object type.

Usage

Application.**GetDefaultColor**(ByVal *ObjectType* As VdObjectType) As IColor

Arguments

- ObjectType

  Type of objects that are to receive the default color. Objects are defined by "VdObjectType Enum" on page 733.

Return Values

The CColor Object.

GetProjectData Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Returns the ProjectData object.

Usage

Application.**GetProjectData**() As IProjectData

Arguments

None

Return Values

As ProjectData. The ProjectData Object.

The ProjectData automation object lets you manage data stored in the application project file and provides some basic file-related information.

Initialize Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Initalizes the application.

Usage

Application.**Initialize**(ByVal *WDIRPath* As String, ByVal *LicensePath* As String) As Long

Arguments

- WDIRPath

  The value of the WDIR environment variable. The WDIR variable specifies the paths that Xpedition Designer searches on initialization for scripts. These paths can exist anywhere on your network.

- LicensePath

  The value of the LM_LICENSE_FILE variable.

Return Values

As Long. Returns True if the initialization was successful. Returns False otherwise.

Description

The application must do some initialization at startup. When started by an automation client, the initialization is delayed until the client calls Initialize so that the client can pass the correct environment to Xpedition Designer. If you start Xpedition Designer you must make this call before accessing any properties or calling any other methods.

MGCDesignerDataManagement Method (Application Object)

Scope: Schematic editor; Embedded client (EDM)

Object: Application Object

Prerequisites: None

Returns the designer data management object.

Usage

Application.**MGCDesignerDataManagement**() As IMGCDesignerDataManagement

Arguments

None

Return Values

As IMGCDesignerDataManagement. The designer data management object.

NewProject Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Creates a new Xpedition Designer project. The new project folder must not already exist.

Usage

Application.**NewProject**(ByVal *ProjectPath* As String, ByVal *CentralLibraryPath* As String, ByVal *ServerName* As String, ByVal *ProjectTemplatePath* As String) As Boolean

Arguments

- ProjectPath

  Specifies the full path to the new project file. Use the format: *..\projectname\projectname.prj*
- CentralLibraryPath

  Specifies the full path to the central library.
- ServerName

  Specifies the iCDB server name or IP address. For a local machine, an empty string could be passed.
- ProjectTemplatePath

  Specifies the path to and name of the project file template. If this argument is an empty string, the default template is used.

Return Values

As Boolean. Returns True if a new project was successfully created; False otherwise.

OpenBlocks Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Returns a collection of Block objects.

Usage

Application.**OpenBlocks**(ByVal *BlockType* As VdDataType) As IVdObjs

Arguments

- BlockType

  Specifies if schematic or symbol blocks are desired. This argument is a VdDataType Enum.

Note:
The only valid values for this argument are VDDT_SCHEMATIC (numeric value 0) and VDDT_SYMBOL (numeric value 1).

Return Values

As IVdObjs. A collection of Block Objects. The returned collection contains one block for each schematic or symbol in the current design.

OpenProject Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Opens a project.

Usage

Application.**OpenProject**(ByVal *ProjectPath* As String)

Arguments

- ProjectPath

  Path to and name of the project file.

OpenURL Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Opens a document at the location specified by a URL in the application.

Usage

Application.**OpenURL**(ByVal *URL* As String)

Arguments

- URL

  String that defines the URL for the document to be opened.

ParamGetMode Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Returns the global mode setting that controls how Xpedition Designer operates.

Usage

*Application.***ParamGetMode**(ByVal *Index* As VdParamMode) As VdOnOff

Arguments

- Index

  Index of the mode of interest.

Return Values

As VdOnOff. The mode value expressed VdOnOff Enum.

ParamGetValue Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Returns the global value setting that controls how Xpedition Designer operates.

Usage

*Application.***ParamGetValue**(ByVal *Index* As VdParamValue) As Long

Arguments

- Index

  Index of the value of interest, as defined by VdParamValue Enum.

Return Values

As Long. A Long that contains the parameter value.

ParamSetMode Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Sets the global mode setting that controls how Xpedition Designer operates.

Usage

*Application.***ParamSetMode**(ByVal *Index* As VdParamMode, ByVal *newValue* As VdOnOff)

Arguments

- index

  Index of the mode of interest, expressed as VdParamMode Enum.
- newValue

  New mode setting, expressed as VdOnOff Enum.

ParamSetValue Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Sets the global value setting that controls how Xpedition Designer operates.

Usage

*Application.***ParamSetValue**(ByVal *Index* As VdParamValue, ByVal *NewValue* As Long)

Arguments

- Index

  Index of the value of interest, expressed as VdParamValue Enum.

- NewValue

  A long that contains the new global setting.

PrintProject Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Prints an entire design, or a portion of a design from the top level down to the level specified.

Note:
This method results in an error if there are any open documents in the design when it is called.

Usage

Application.**PrintProject**(ByVal *DesignName* As String, ByVal *LevelStrings* As String, ByVal *PrintSymbols* As Boolean, ByVal *ShowOrderDialog* As Boolean, ByVal *PPOFile* As String)

Arguments

- DesignName

  Name of the schematic which is the top of the design.
- LevelStrings

  Level string(s) that specify the hierarchical level(s) at which to stop traversing into composites. For example, "STD VHDL" would stop at the primitive, STD, and VHDL levels of hierarchy.
- PrintSymbols

  If True, symbols will be printed in addition to schematics. If False, only schematics will be printed.
- ShowOrderDialog

  If True, a dialog box is displayed in which you specify which schematics to print, and the order in which they are printed. If False, the all schematics in the design (down to the specified level of hierarchy) are printed in the default order.
- PPOFile

  File specification of a Project Print Order file containing print order information. If specified as the empty string ("""), a default order is used. A PPO file can be created by clicking Save in the Project Print Order dialog box. If ShowOrderDialog is True, this parameter determines the initial settings. If ShowOrderDialog is False, this parameter determines what gets printed. Example: "c:/projects/counter.ppo".

Examples

Print the project starting at "top", down to LEVEL=STD, without symbols, without a print order dialog, in the default order.

```
PrintProject "top", "STD", False, False, ""
```

ProductionLibraryLimitationsChanged Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Updates the Variant Manager Settings dialog box when the production library limitation is modified.

Usage

Application.**ProductionLibraryLimitationsChanged**()

Arguments

None

Return Values

Nothing

PushPath Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Opens a schematic sheet by pushing through the components of a path.

Usage

Application.**PushPath**(ByVal *Top* As String, ByVal *HierPath* As String, ByVal *SheetNumber* As String) As Boolean

Arguments

- Top

  Name of the schematic that is the top of the design. This argument is a string with a specific format, resembling a path specification:

  ```
  [library_name:]sch\name.sheetnumber
  ```

  library_name is the (optional) alias assigned to the library containing this object. *sch*\ specifies that the sheet is a schematic. *name.sheetnumber* is the name and sheet number of the schematic.For example:

  ```
  My_comps:sch\foo.1
  ```

  Note:
  In C++, you must escape the \ character as \\.

- HierPath

  Hierarchical path specification. Defines the path from the top-level schematic, specified by the top argument, to the target schematic. It's a sequence of UIDs separated by the \ character. The schematic representing the final component is opened by pushing along the path specified by the other components. An example path could be $1I2\$1I1\$1I1 (or "$1I2\\$1I1\\$1I1" in C++).

- SheetNumber

  Contains either the sheet name or the actual sheet number of interest. For example, in a schematic sheet, "Schematic1.cpu", the *SheetNumber* argument would be "cpu". A *SheetNumber* value of “-1” indicates that the application open the default sheet.

Return Values

As Boolean. True - the path was found and the schematic opened (success). False - the path could not be found and/or the schematic could not be opened (failure).

Description

The schematic is opened based on its relationship to the top of the design. Only the target schematic is opened, not all the schematics along the path.

Query Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Returns a collection of objects based on type.

Usage

Application.**Query**(ByVal *Flags* As VdObjectTypeMask, ByVal *Selected* As VdAllOrSelected) As IVdObjs

Arguments

- Flags

  Masks which may be combined to filter the objects selected by this method. This argument is of type VdObjectTypeMask Enum.

- Selected

  Determines whether to consider all objects or just selected objects. This argument is of type VdAllOrSelected Enum.

Return Values

As IVdObjs. This is a collection of objects that have been filtered by the *Flags* argument.

Description

The Query method is very valuable for locating and examining all objects of a particular type. You can perform the query on all objects or on a (filtered) set of objects.

Examples

Find all the selected components in all sheets in memory.

```
For Each Comp In App.Query(VDM_COMP, VD_SELECTED)
 MsgBox "Component UID=" & Comp.UID
Next
```

QueryPages Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Returns a collection of block objects.

Usage

Application.**QueryPages**(ByVal *Name* As String) As IVdObjs

Arguments

- Name

  Name of the schematic (do not include the extension).

Return Values

As IVdObjs. A collection of Block Objects. The collection contains one block for each sheet of the schematic.

Quit Method (Application Object)

Object: Application Object

Prerequisites: None

Terminates the application.

Linux:
Do not use this method in scripts that run at startup. This method will cause an error when called before the Viewdraw application has fully initialized.

Usage

Application.**Quit**()

Arguments

None

Description

Call this method last method if your application started Xpedition Designer. Release all automation interfaces (except the application interface) prior to this method call. This has the same effect as choosing Exit from the File menu.

RunISE Method (Application Object)

Scope: Schematic editor - Embedded symbol editor (ESE)

Object: Application Object

Prerequisites: None

Opens an existing non-local symbol in the embedded symbol editor. Otherwise, creates a new empty symbol with a given name.

Usage

Application.**RunISE**(ByVal *Path* As String) As Boolean

Arguments

- Path

  String. A string that contains one of the following:

  - The path to the project file and the symbol (Netlist flow)
  - The path to the library, partition, and name of the symbol (Library Manager mode)

Return Values

Boolean. Returns True if the symbol opened in ESE successfully. Otherwise, returns False.

Examples

```
Set app = CreateObject("viewdraw.Application")
Scripting.AddTypeLibrary("Viewdraw.Application")

’Opens symbol cap.1 from file – Netlist Flow
run_ESE="-prj c:\MyProjects\myNetlistProject.prj" _
  + "-file c:\MyLibs\Discrete\sym\cap.1"

app.RunISE(run_ESE)

’Opens symbol cap.1 from partition Discrete from Central Library
 ’SampleLib2007
run_ESE="-lmc C:\MentorGraphics\EEVX.2.10\SDD_HOME\standard\" _
  + "examples\SampleLib2007\SymbolLibs -partition Discrete -- cap.1"

app.RunISE(run_ESE)
```

SchematicSheetDocuments Method (Application Object)

Object: Application Object

Prerequisites: None

Returns a collection of SchematicSheetDocument objects.

Usage

Application.**SchematicSheetDocuments**() As IVdSchematicSheetDocuments

Arguments

None

Return Values

As IVdSchematicSheetDocuments. A collection of SchematicSheetDocument Objects. SchematicSheetDocument is an Automation object that represents a sheet of a schematic.

SelectPath Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Opens a schematic and selects objects based on a path from the top of the design.

Usage

Application.**SelectPath**(ByVal *FileName* As String, ByVal *HierPath* As String, ByVal *SheetNumber* As String, ByVal *Type* As Long, ByVal *AddSelected* As Boolean, ByVal *SearchBus* As Boolean) As Boolean

Arguments

- FileName

  Name of the document that is the top of the design. This argument is a string with a specific format, resembling a path specification:

  ```
  [library_name:]name.sheetnumber
  ```

  library_name is the (optional) alias assigned to the library containing this object. You need not specify the library_name when opening the top level sheet of the schematic. *name.sheetnumber* is the name and sheet number of the schematic. For example:

  ```
  dxApp.SelectPath "Schematic1", "\$1I9", "", 0, True, False
  ```

  These paths are often included in error and warning messages. Only the target document is opened, not all the documents along the path.

  Note:
  In C++ , you must escape the \ character as \\.

- HierPath

  Hierarchical path specification. Defines the path from the top-level document specified by FileName to the target item. It has the format of:

  ```
  instance_name\instance_name\...\instance_name
  ```

  For example, $1I2\$1I1\$1I1 or the C++ string would be "$1I2\\$1I1\\$1I1"

- SheetNumber

  Contains either the actual sheet number of interest or, in most cases, “” which indicates that the application open the default sheet.

- Type

  Specifies which type of object to select. This value is ignored if the *HierPath* argument is specified using the internal unique identifiers (as shown above). If the path is defined using labels, then the *Type* value must be one of the following:

    - **0** — selects a component.
    - **1** — selects a net.
- AddSelected

  Specifies how to handle previously selected objects:

    - **True** — add to currently selected objects.
    - **False** — replace current selection.
- SearchBus

  This value is ignored unless no matching net labels are found.

    - **True** — Look for label in expanded bus labels.
    - **False** — Only look for the label on nets.

Return Values

As Boolean:

- **True** — The path was found and at least one object was selected.
- **False** — The application could not fine a path or an object.

SelectPathCompPin Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Opens a schematic and selects a component pin based on a path from the top of the design.

Note:
The method only opens the target document, and not all the documents along the path.

Usage

Application.**SelectPathCompPin**(ByVal *FileName* As String, ByVal *HierPath* As String, ByVal *SheetNumber* As String, ByVal *PinNumber* As String, ByVal *SelectType* As Long) As Boolean

Arguments

- FileName

  Name of the document which is the top of the design. This argument is a string with a specific format, resembling a path specification:

  ```
  [library_name:]sch\name.sheetnumber
  ```

  library_name is the (optional) alias assigned to the library containing this object. *sch* specifies that the sheet is a schematic. *name.sheetnumber* is the name and sheet number of the schematic. For example:

  ```
  My_comps:sch\foo.1
  ```

  Note:
  In C++, you must escape the \ character as \\.

- HierPath

  Hierarchical path specification. Defines the path from the top-level document specified by FileName to the target item. It has the format of:

  ```
  instance_name\instance_name\...\instance_name
  ```

  For example, $1I2\$1I1\$1I1 or the C++ string would be

  ```
  "$1I2\\$1I1\\$1I1"
  ```

- SheetNumber

  Contains either the actual sheet number of interest or, in most cases, “” which indicates that the application open the default sheet.

- PinNumber

  String that matches the pin number attribute assigned to the pin.

- SelectType

  Specifies which type of object to select. This value is ignored if *HierPath* is specified using the internal unique identifiers (as shown above). If the path is defined using labels, then the *SelectType* value must be one of the following:

  - **0** — Selects a symbol pin name.
  - **1** — Selects a component pin

Return Values

As Boolean:

- **True** — A pin was selected.
- **False** — No pin was found or selected.

SetDefaultColor Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Sets the new color (in RGB format) for specified object type.

Note:
This method only applies to nets, components, attributes, and labels. Trying to use this method to set the default color for any other object type results in an error.

Usage

Application.**SetObjectColor**(ByVal *ObjectType* As VdObjectType, ByVal *Color* as IColor)

Arguments

- ObjectType

  Type of objects for which to set the new color. Objects are defined by "VdObjectType Enum" on page 733.

  Only the following VdObject enumerated types are valid as arguments for this method:

  - VDTS_LINE (numerical value 0)
  - VDTS_BOX (numerical value 1)
  - VDTS_TEXT (numerical value 2)
  - VDTS_CIRCLE (numerical value 3)
  - VDTS_ARC (numerical value 4)
  - VDTS_NET (numerical value 5)
  - VDTS_ATTRIBUTE (numerical value 6)
  - VDTS_COMPONENT (numerical value 7)
  - VDTS_LABEL (numerical value 8)
  - VDTS_PIN (numerical value 9)
  - VDTL_ANNOTATION (numerical value 1036)
  - VDTL_SELECTION (numerical value 1037)

  - VDTL_BACKGROUND (numerical value 1038)
  - VDTL_BORDER (numerical value 1040)
  - VDTL_HIGHLIGHT (numerical value 1041)
- Color

  New color in RGB format (CColor Object).

SetRedraw Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Allows or prevents changes from being drawn.

Usage

Application.**SetRedraw**(ByVal *Redraw* As Boolean)

Arguments

- Redraw

  Specifies whether to enable or disable drawing

  True - Enable drawing of the window as changes are made. False - Disable drawing of the window as changes are made.

Examples

You can use this method to prevent the application from drawing changes for a period of time.

For example, a script could call SetRedraw(FALSE), perform operations which would normally update the screen, then call SetRedraw(TRUE) to see all the changes at once. Calls to SetRedraw(FALSE) and SetRedraw(TRUE) must be paired and can be nested. The final call to SetRedraw(TRUE) automatically cause the views to be refreshed.

StartMigration Method (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Allows you to perform a project migration from EXP2005.1 to EXP 2007.

Usage

Application.**StartMigration**(ByVal *ProjectPath* As String, ByVal *CentralLib* As String) As Boolean

Arguments

- ProjectPath

  The path to and name of the project file (*.dproj*).
- CentralLib

  The path to and name of the central library.

Return Values

As Boolean. True - the migration was completed(success). False - the migration could not be completed (failure).

ActiveDocument Property (Application Object)

Scope: Schematic editor

Object: Application Object

Access: Read-Only

Prerequisites: None

Returns the ActiveDocument collection.

Usage

Application.**ActiveDocument**

Arguments

None

Return Values

Object. A document object (HDLSourceDocument Object or SchematicSheetDocument Object).

Description

The ActiveDocument may be one of:

- **schematic** — SchematicSheetDocument
- **symbol** — SchematicSheetDocument
- **text** — HDLSourceDocument

ActiveView Property (Application Object)

Scope: Schematic editor

Object: Application Object

Access: Read-Only

Prerequisites: None

Returns the ActiveView collection for the currently activated schematic or symbol.

Usage

Application.**ActiveView**

Arguments

None

Return Values

IVdView. The View Object for the currently activated Schematic or Symbol window.

Description

Xpedition Designer has the concept of an active view. This is the active document window if Xpedition Designer is visible. The view is used to control zooming, appearance and supports other operations. This property contains an automation interface pointer to Xpedition Designer's currently active view. If no document is open there will not be an active view and this call will return NULL.

ActiveView is a direct way to go from the Application Object to the currently active View. It returns a view object. The common mistake that programmers make with this interface is to forget that if no documents are open then there is no active view and this call will return NULL. In VBSCRIPT the way to test for this condition is by using the Is Nothing statement.

Examples

Test to see if there is a View.

```
If ActiveView Is Nothing Then
    MsgBox "No View present"
Else
 MsgBox "View present"
End If
```

Addins Property (Application Object)

Object: Application Object

Access: Read-Only

Prerequisites: None

Allows access to an Addins collection.

Usage

Application.**Addins**

Arguments

None

Return Values

Object. A collection of Addins.

CommandBars Property (Application Object)

Scope: Schematic editor

Object: Application Object

Access: Read-Only

Prerequisites: None

Returns a collection of CommandBar objects.

Usage

Application.**CommandBars** As CommandBarSvr.ICommandBars

Arguments

None

Return Values

CommandBarSvr.ICommandBars. Refer to CommandBar Server in the Common Automation Manual.

Description

CommandBar objects are the primary object to use when programming menus and toolbars (refer to Figure 3: CommandBarServer Data Model).

Figure 3. CommandBarServer Data Model

CommandLineArguments Property (Application Object)

Scope: Schematic editor

Object: Application Object

Access: Read-Only

Prerequisites: None

Returns the command line arguments that were used to call the application.

Usage

Application.**CommandLineArguments**

Arguments

None

Return Values

String. A string of the arguments that were used to call the application.

Interactive Property (Application Object)

Scope: Schematic editor

Object: Application Object

Access: Read/Write

Prerequisites: None

Controls the response of the application to activity on its user interface.

Usage

Application.**Interactive** = True | False

Arguments

None

Return Values

True | False. True - Xpedition Designer is in interactive mode. False - Xpedition Designer is not in interactive mode.

Description

The application is normally interactive (True) and responds to user input. Setting Interactive to False ignores all mouse clicks, keyboard hits, and other user actions. In effect, locking out the user while critical operations are proceeding. When Interactive is False, only the automation script and the operating system can affect the application.

QueueSelectEvents Property (Application Object)

Scope: Schematic editor

Object: Application Object

Access: Read/Write

Prerequisites: None

Sets or disables event queuing.

Usage

Application.**QueueSelectEvents** = Long

Arguments

None

Return Values

Long. When this property is set to True, the application posts each Select event to a queue without execution. When set to False, the application executes the Select event for each queued block, if any.

Put another way, true enables queuing of Select events, while false does not queue the Select events, but executes them immediately.

ShellCmd Property (Application Object)

Scope: Schematic editor

Object: Application Object

Access: Read-Only

Prerequisites: None

Returns the ShellCmd collection, which allows you to execute programs.

Usage

Application.**ShellCmd**

Arguments

None

Return Values

Object. The return type for this property.

SilentMode Property (Application Object)

Scope: Schematic editor

Object: Application Object

Access: Read/Write

Prerequisites: None

Returns or sets the silent mode status of the application.

Note:
When SilentMode is set to VDSM_ALL, the user is not given the option to discard core dump files if a crash occurs. In that case, core dump files are automatically written to the WDIR directory.

Usage

Application.**SilentMode** = VdSilentMode

Arguments

None

Return Values

Object. A value which determines if the application is in silent mode or not, expressed as VdSilentMode Enum.

When the application is in silent mode, no dialog boxes or popups are displayed. Therefore, no user intervention is required when performing a batch process.

SourceDocuments Property (Application Object)

Scope: Schematic editor

Object: Application Object

Access: Read-Only

Prerequisites: None

Returns a collection of HDLSourceDocument objects.

Usage

Application.**SourceDocuments**

Arguments

None

Return Values

IHDLSourceDocuments. An HDLSourceDocuments Collection.

An HDLSourceDocument Object is an automation object that represents an HDL source file that has been opened in Xpedition Designer.

StatusBarText Property (Application Object)

Scope: Schematic editor

Object: Application Object

Access: Read/Write

Prerequisites: None

Returns or sets the text that is displayed in the application status bar.

Usage

Application.**StatusBarText** = String

Arguments

None

Return Values

String. A string of text that appears in the status bar.

Version Property (Application Object)

Scope: Schematic editor

Object: Application Object

Access: Read-Only

Prerequisites: None

Returns the version number of the application that appears when you choose About Xpedition Designer on the Help menu.

Usage

Application.**Version**

Arguments

None

Return Values

String. A string relating the version number of the application that is currently in use.

Examples

This syntax causes a message box to appear with the current Xpedition Designer version number.

```
MsgBox Application.Version
```

Visible Property (Application Object)

Scope: Schematic editor

Object: Application Object

Access: Read/Write

Prerequisites: None

Returns or sets the visibility of the application.

Note:
It may be helpful to display a message to the user that the application is invisible to enable rapid Automation drawing during this process.

Usage

Application.**Visible** = True | False

Arguments

None

Return Values

True | False. True - Xpedition Designer is visible. False - Xpedition Designer is invisible.

Description

Xpedition Designer is normally visible (that is, the Visible property is “true”). However, when first started by an automation script, the Visible property is “false”. Changing this property to “True” is a standard initialization step.

You may want to make Xpedition Designer invisible when doing large amounts of drawing, as this can speed performance. Xpedition Designer does not actually draw the data until turned visible and therefore the drawing operations run much faster (by a factor of up to 10).

ActivateView Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs when the View child window becomes active.

Usage

Sub **Application_ActivateView**()

Arguments

None

Description

This event is executed when a child window becomes active (receives focus), not when the Application becomes active.

Note:
When this event is executed, the context has not yet been set, so GetTopLevelDesignName Method (View Object) and TopBlock Property (View Object) might return incorrect information. To avoid that problem consider using ActivateView2 Event (Application Object) instead of this event.

ActivateView2 Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs when the View child window becomes active (receives focus), not when the application becomes active.

Usage

Sub **Application_ActivateView2**(ByVal *View* As IVdView)

Arguments

- View

  This is the View Object that is becoming active.

AfterDocumentOpened Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs after a document is opened and drawn.

Usage

Sub **Application_AfterDocumentOpened**(ByVal *DocumentType* As VdDataType, ByVal *LibraryAlias* As String, ByVal *Name* As String)

Arguments

- DocumentType

  Specifies the document type. This argument takes the form of VdDataType Enum.
- LibraryAlias

  This is a string that corresponds to the library alias.
- Name

  This is the name of the document.

AfterPrintProject Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs after a print project is completed.

Usage

Sub **Application_AfterPrintProject**()

Arguments

None

AfterSheetRead Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs after a sheet is read into memory.

Usage

Sub Application_**AfterSheetRead**(ByVal *DocumentType* As VdDataType, ByVal *LibraryAlias* As String, ByVal *Name* As String)

Arguments

- DocumentType

  Specifies the document type. This argument takes the form of VdDataType Enum.
- LibraryAlias

  This is a string that corresponds to the library alias.
- Name

  This is the name of the document.

AfterSheetReRead Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs after a sheet has been reread.

Usage

Sub Application**_AfterSheetReRead**(ByVal *Block* As IVdBlock)

Arguments

- Block

  Specifies the Block Object that has been reread.

BeforeDocumentOpened Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs before a document is opened and drawn.

Usage

Private Sub **Application_BeforeDocumentOpened**(ByVal *DocumentType* As VdDataType, ByVal *LibraryAlias* As String, ByVal *Name* As String)

Arguments

- DocumentType

  Specifies the document type. This argument is of the form VdDataType Enum.
- LibraryAlias

  This is a string that corresponds to the library alias.
- Name

  This is the name of the document.

BeforePrintProject Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs before a print project is executed.

Usage

Sub **Application_BeforePrintProject**(ByVal *FileList* As String)

Arguments

- FileList

  This is a comma-delimited list of files to be printed.

BeforeProjectChanged Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs before the project is changed. That is, the event occurs before a opening another project.

Usage

Sub **Application_BeforeProjectChanged()**

Arguments

None

BlockLocked Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs when you change the state of a schematic block from read-only to read/write, using the “Click to Edit” button.

Usage

Sub **Application_BlockLocked**(ByVal *Block* As IVdBlock)

Arguments

- Block

  Specifies the locked Block Object that is changing to read/write.

BlockModified Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs every time a Block Object is modified.

Usage

Sub **Application_BlockModified**(ByVal *Block* As IVdBlock)

Arguments

- Block

  The name of the Block Object that has been modified.

Description

See Block Object for more information.

CreateObject Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs whenever a new object is created. Use this event to stipulate other events, such as verifying that a component is labeled, has correct attributes, and so on.

Usage

Sub **Application_CreateObject**(ByVal *ReasonFlag* As VdCreateTime, ByVal *ObjectType* As VdObjectType, ByVal *Block* As IVdBlock, ByVal *Object* As Object)

Arguments

- ReasonFlag

  Specifies that the event occurs either before or after the object is placed. This argument takes the form of VdCreateTime Enum.

- ObjectType

  Specifies the type of object being created in the form of VdObjectType Enum.

- Block

  The block in which the object is being created.

- Object

  The dispatch interface of the object that was just created. You must cast it to the correct type by using the ObjectType parameter.

Examples

Add a label when new components are created.

```
Sub Application_CreateObject(ByVal ReasonFlag,  ByVal ObjectType,  ByVal
 Block,  ByVal Object)
If ObjectType = VDTS_COMPONENT Then
Object.AddLabel "LABEL", Object.GetLocation().X, Object.GetLocation().Y
    End If
End Sub
```

DeactivateView Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs when a child window is deactivated. This event is not triggered when the Application is deactivated.

Usage

Sub **Application_DeactivateView**()

Arguments

None

Examples

When a child window is closed or another child becomes active, the Deactivate View event is triggered for the original child window.

DeactivateView2 Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs when a child window is deactivated. This event is not triggered by deactivating the Application.

Usage

Sub **Application_DeactivateView2**(ByVal *View* As IVdView)

Arguments

- View

  The View Object that is being deactivated.

Examples

When a child window is closed or another child becomes active, the Deactivate View event is triggered for the original child window.

Delete Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs whenever an object is deleted.

Usage

Sub **Application_Delete**()

Arguments

None

DocumentClose Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs whenever a SchematicSheetDocument Object is closed.

Usage

Sub **Application_DocumentClose**(ByVal *Doc* As IVdDoc)

Arguments

- Doc

  The SchematicSheetDocument Object that is being closed.

Description

See SchematicSheetDocument Object for more information.

LockRequest Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs once before modifying a Block Object.

Usage

Sub **Application_LockRequest**(ByVal *Block* As IVdBlock, ByVal *Success* As IPredicate)

Arguments

- Block

  Specifies the Block Object that is being locked.

- Success

  This argument is set by the event handler to prevent/allow modification.

  True - Allows the application to modify the block. False - prevents the application from modifying the block, indicating that the block has already been locked.

MouseMoved Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

This event allows you to track mouse movements and occurs when the mouse is moved in a View.

Usage

Sub **Application_MouseMoved**(ByVal *Flags* As Long, ByVal *X* As Long, ByVal *Y* As Long)

Arguments

- Flags

  Indicates whether or not various virtual keys are active. This parameter can be any combination of the following values:

  - **1** — Set if the left mouse button is down.
  - **2** — Set if the right mouse button is down.
  - **4** — Set if the SHIFT key is down.
  - **8** — Set if the CTRL key is down
  - **16** — Set if the middle mouse button is down
- X

  X coordinate of the mouse position. This value is expressed in the application coordinate system.
- Y

  Y coordinate of the mouse position. This value is expressed in the application coordinate system.

PaintRegion Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs when a View is painted.

Usage

Sub **Application_PaintRegion**(ByVal *View* As IVdView, ByVal *LowerLeftx* As Long, ByVal *LowerLefty* As Long, ByVal *UpperRightx* As Long, ByVal *UpperRighty* As Long)

Arguments

- View

  Specifies the name of the View being painted.
- LowerLeftx

  X coordinate of the lower left hand corner of the clip rectangle. This value is expressed in the application coordinate system.
- LowerLefty

  Y coordinate of the lower left hand corner of the clip rectangle. This value is expressed in the application coordinate system.
- UpperRightx

  X coordinate of the upper right hand corner of the clip rectangle. This value is expressed in the application coordinate system.
- UpperRighty

  Y coordinate of the upper right hand corner of the clip rectangle. This value is expressed in the application coordinate system.

Description

The View being painted, and the dimensions of the clip rectangle are passed as parameters. Use this method as a hook into the Viewport object.

PrintFile Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs when a file is printed.

Usage

Sub **Application_PrintFile**(ByVal *BlockName* As String, ByVal *Path* As String, ByVal *TopBlock* As String)

Arguments

- BlockName

  Specifies the block name.
- Path

  Specifies the current hierarchical path.
- TopBlock

  Specifies the top level block name.

ProductionLibraryLimitationsChanged Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs when the production library limitation is changed.

Usage

Sub **Application_ProductionLibraryLimitationsChanged**()

Arguments

None

ProjectChanged Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs after a project is changed and then re-opened.

Usage

Sub Application**_ProjectChanged**(ByVal *ProjectData* As IProjectData)

Arguments

- ProjectData

  Specifies the ProjectData Object that triggers the event.

ProjectClosed Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs after a project is closed.

Usage

Private Sub Application**_ProjectClosed**()

Arguments

None

Select Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs when an object is selected or unselected. You can use this event to access such things as crossprobing.

Usage

Sub **Application_Select**(ByVal *SelectionType* As VdSelectionType, ByVal *Block* As IVdBlock)

Arguments

- SelectionType

  This argument specifies the selection type. It takes the form of VdSelectionType Enum.
- Block

  This argument specifies the Block Object that contains the object that is being selected.

Examples

Display the ID of the selected component.

```
Sub Application_Select(SelectionType, Block)
    If SelectionType = VDSELECT_NOTIFY Then
  For Each obj In ActiveView.Query(VDM_COMP, VD_SELECTED)
      MsgBox obj.UID
  Next
    End If
End Sub
```

Shutdown Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs when Xpedition Designer shuts down.

Usage

Sub **Application_Shutdown**()

Arguments

None

SourceDocumentSave Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs when a source document (VHDL, Verilog, or Spice) is saved.

Usage

Sub **Application_SourceDocumentSave**(ByVal *DocumentName* As String)

Arguments

- DocumentName

  This argument specifies the name of the file being saved.

SourceFileModified Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs when an existing source document (VHDL, Verilog, or Spice) is modified and saved.

Usage

Sub **Application_SourceFileModified**(ByVal *FileName* As String)

Arguments

- FileName

  This argument specifies the name of the file being saved.

Startup Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs when Xpedition Designer starts.

Usage

Sub **Application_Startup**()

Arguments

None

SymbolPreviewed Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs when the symbol previewer displays a new symbol.

Usage

Sub **Application_SymbolPreviewed**(ByVal *SymbolBlock* As IVdBlock)

Arguments

- SymbolBlock

  Specifies the symbol Block Object that is being displayed.

Unlock Event (Application Object)

Scope: Schematic editor

Object: Application Object

Prerequisites: None

Occurs when a block is being unlocked (the application saves a block or closes a block without saving it).

Usage

Sub **Application_Unlock**(ByVal *Block* As IVdBlock)

Arguments

- Block

  The name of the Block Object that is being unlocked.

Arc Object

This object represents the Xpedition Designer arc graphical object.

The following tables list methods and properties of the Arc object with links to the respective reference pages:

Table 6. Arc Object Methods and Properties

Method or Property	Description
GetLocation Method (Arc Object)	Returns the specified arc coordinate.
GetObjectColor Method (Arc Object)	Gets the color in which the arc is drawn.
IsColorAutomatic Method (Arc Object)	Determines if the arc has an automatic color set.
SetAutomaticColor Method (Arc Object)	Sets or unsets the color of an arc object as the automatic color for arcs.
SetLocation Method (Arc Object)	Modifies an arc position by specifying three coordinates of the new arc location.
SetObjectColor Method (Arc Object)	Sets the color when drawing the arc.
Application Property (Arc Object)	Returns the Application Object.
LineStyle Property (Arc Object)	Returns or sets the line style for the arc.
Parent Property (Arc Object)	Returns the parent Attribute Object in which this arc was created.
Selected Property (Arc Object)	Sets the arc selection status.
Type Property (Arc Object)	Returns the arc object type, as VdObjectType Enum.

GetLocation Method (Arc Object)

Scope: Schematic editor

Object: Arc Object

Prerequisites: None

Returns the specified arc coordinate.

Usage

Arc.**GetLocation**(ByVal *WhichPoint* As VdArcPoint) As IVdPoint

Arguments

- WhichPoint

  This argument specifies which arc point coordinate to return. It takes the form of VdArcPoint Enum.

Return Values

As IVdPoint. The arc coordinate (measured in 100ths of an inch).

GetObjectColor Method (Arc Object)

Scope: Schematic editor

Object: Arc Object

Prerequisites: None

Gets the color in which the arc is drawn.

Usage

Arc.**GetObjectColor**() as IColor

Arguments

None

Return Values

As Color. The CColor Object.

IsColorAutomatic Method (Arc Object)

Scope: Schematic editor

Object: Arc Object

Prerequisites: None

Determines if the arc has an automatic color set.

Usage

Arc.**IsColorAutomatic**() As Boolean

Arguments

None

Return Values

As Boolean. True - there is an automatic color set for the arc. False - no automatic color is set for the arc.

For more information, see "SetAutomaticColor Method (Arc Object)" on page 154.

SetAutomaticColor Method (Arc Object)

Scope: Schematic editor

Object: Arc Object

Prerequisites: None

Sets or unsets the color of an arc object as the automatic color for arcs.

Usage

Arc.**SetAutomaticColor**(byVal *bAutomatic* as Boolean)

Arguments

- bAutomatic.

  True - sets the automatic color for that object. False - unsets the automatic color for that object.

  If an object has automatic color, as determined by the IsColorAutomatic Method (Arc Object), the actual color of the object is the default for this type of object.

SetLocation Method (Arc Object)

Scope: Schematic editor

Object: Arc Object

Prerequisites: None

Modifies an arc position by specifying three coordinates of the new arc location.

Note:
All coordinates are measured in 100ths of an inch.

Usage

Arc.**SetLocation**(ByVal *X1* As Long, ByVal *Y1* As Long, ByVal *X2* As Long, ByVal *Y2* As Long, ByVal *X3* As Long, ByVal *Y3* As Long)

Arguments

- X1

  The X coordinate of the first point.
- Y1

  The Y coordinate of the first point.
- X2

  The X coordinate of the second point.
- Y2

  The Y coordinate of the second point.
- X3

  The X coordinate of the third point.
- Y3

  The Y coordinate of the third point.

SetObjectColor Method (Arc Object)

Scope: Schematic editor

Object: Arc Object

Prerequisites: None

Sets the color when drawing the arc.

Note:
Setting the color with this method resets the automatic flag (see SetAutomaticColor Method (Arc Object).)

Usage

Arc.**SetObjectColor**(ByVal *newColor* as IColor)

Arguments

- NewColor.

  The color assigned to the arc. The new color is assigned as a Color object, as described in "CColor Object" on page 256.

Application Property (Arc Object)

Scope: Schematic editor

Object: Arc Object

Access: Read-Only

Prerequisites: None

Returns the Application Object.

Usage

*Arc.***Application**

Arguments

None

Return Values

IVdApp. The Application Object.

LineStyle Property (Arc Object)

Scope: Schematic editor

Object: Arc Object

Access: Read/Write

Prerequisites: None

Returns or sets the line style for the arc.

Usage

Arc.**LineStyle** = VdLineStyle

Arguments

None

Return Values

VdLineStyle. The return/set type for this property. This takes the form of VdLineStyle Enum.

Parent Property (Arc Object)

Scope: Schematic editor

Object: Arc Object

Access: Read-Only

Prerequisites: None

Returns the parent Attribute Object in which this arc was created.

Usage

Arc.**Parent**

Arguments

None

Return Values

IVdBlock. The return type for this property.

See Attribute Object for more information.

Selected Property (Arc Object)

Scope: Schematic editor

Object: Arc Object

Access: Write-Only

Prerequisites: None

Sets the arc selection status.

Usage

Arc.**Selected** = True | False

Arguments

None

Return Values

True | False. True - Selects the arc. False - Deselects the arc.

Type Property (Arc Object)

Scope: Schematic editor

Object: Arc Object

Access: Read-Only

Prerequisites: None

Returns the arc object type, as VdObjectType Enum.

Usage

Arc.**Type**

Arguments

None

Return Values

VdObjectType. The return type for this property, taking the form of VdObjectType Enum.

Attribute Object

This object represents an attribute of an object (a net, component, block, or other object) on a schematic.

The following table lists methods and properties of the Attribute object with links to the respective reference pages:

Table 7. Attribute Object Methods and Properties

Method or Property	Description
Delete Method (Attribute Object)	Deletes the attribute.
DeleteInstanceValue Method (Attribute Object)	Deletes the instance value from the attribute.
GetLocation Method (Attribute Object)	Returns the coordinates of the attribute taking the form of a Point Object.
GetOatFull Method (Attribute Object)	Gets an instance value from within a particular context in the design.
GetObjectColor Method (Attribute Object)	Gets the color in which the attribute is drawn.
IsColorAutomatic Method (Attribute Object)	Determines if the attribute has an automatic color set.
SetAutomaticColor Method (Attribute Object)	Sets or unsets the color of an attribute object as the automatic color for attributes.
SetLocation Method (Attribute Object)	Specifies the location for an attribute.
SetObjectColor Method (Attribute Object)	Sets the color in which the Attribute is drawn.
Application Property (Attribute Object)	Returns the Application Object.
Child Property (Attribute Object)	Returns the child Block Object of the attribute.
EitherValue Property (Attribute Object)	Returns or sets the value of the attribute.
Font Property (Attribute Object)	Returns or sets the font that is used to draw the attribute.
InstanceValue Property (Attribute Object)	Returns or sets the instance (OAT) value for an attribute.
Name Property (Attribute Object)	Returns the name of the attribute.
Orientation Property (Attribute Object)	Returns or sets the attribute orientation.
Origin Property (Attribute Object)	Returns or sets one of the nine origins that is used for this attribute.
Parent Property (Attribute Object)	Returns the parent Block Object for the attribute.
Selected Property (Attribute Object)	Sets the selection status of the attribute.
Size Property (Attribute Object)	Returns or sets the text size for the attribute.

Table 7. Attribute Object Methods and Properties (continued)

Method or Property	Description
TextString Property (Attribute Object)	Returns or sets the string associated with an attribute.
Type Property (Attribute Object)	Returns the attribute type.
Value Property (Attribute Object)	Returns or sets the attribute's regular value even if the attribute has an instance (OAT) value.
Visible Property (Attribute Object)	Returns or sets the visibility of the attribute.

Delete Method (Attribute Object)

Scope: Schematic editor

Object: Attribute Object

Prerequisites: None

Deletes the attribute.

Usage

Attribute.**Delete**()

Arguments

None

Description

After deleting the attribute, accessing properties or methods of the attribute object is undefined and could result in an exception.

DeleteInstanceValue Method (Attribute Object)

Scope: Schematic editor

Object: Attribute Object

Prerequisites: None

Deletes the instance value from the attribute.

Usage

Attribute.**DeleteInstanceValue**()

Arguments

None

GetLocation Method (Attribute Object)

Scope: Schematic editor

Object: Attribute Object

Prerequisites: None

Returns the coordinates of the attribute taking the form of a Point Object.

Note:
All coordinates are measured in 100ths of an inch.

Usage

Attribute.**GetLocation**() As IVdPoint

Arguments

None

Return Values

As IVdPoint. The coordinates of the attribute.

See Point Object for more information.

Examples

This example displays a message box with the coordinates of an attribute.

```
MsgBox "Attribute is at X=" & Attr.GetLocation().X & " Y=" &
 Attr.GetLocation().Y
```

GetOatFull Method (Attribute Object)

Scope: Schematic editor

Object: Attribute Object

Prerequisites: None

Gets an instance value from within a particular context in the design.

Usage

Attribute.**GetOatFull**(ByVal *toplevel* As String, ByVal *pathvalue* As String, ByVal *full_or_not* As Long) As String

Arguments

- toplevel

  This is the name of the top level design.
- pathvalue

  This is the path to the instance value that is to be retrieved.
- full_or_not

  This argument specifies whether or not the entire path to the OAT is used.

  True - the entire path is used. False - only the end object of the OAT value is used.

Return Values

As String. The OAT (and, optionally the path) indicated by the arguments.

GetObjectColor Method (Attribute Object)

Scope: Schematic editor

Object: Attribute Object

Prerequisites: None

Gets the color in which the attribute is drawn.

Usage

Attribute.**GetObjectColor**() As IColor

Arguments

None

Return Values

As Color. The color of the attribute.

IsColorAutomatic Method (Attribute Object)

Scope: Schematic editor

Object: Attribute Object

Prerequisites: None

Determines if the attribute has an automatic color set.

Usage

Attribute.**IsColorAutomatic**() As Boolean

Arguments

None

Return Values

As Boolean. True - there is an automatic color set for the attribute. False - no automatic color is set for the attribute.

For more information, see "SetAutomaticColor Method (Attribute Object)" on page 170.

SetAutomaticColor Method (Attribute Object)

Scope: Schematic editor

Object: Attribute Object

Prerequisites: None

Sets or unsets the color of an attribute object as the automatic color for attributes.

Usage

Attribute.**SetAutomaticColor**(byVal *bAutomatic* as Boolean)

Arguments

- bAutomatic

  True - sets the automatic color for the attribute. False - unsets the automatic color for that attribute.

  If an object has automatic color, as determined by the IsColorAutomatic Method (Attribute Object), the actual color of the object is the default for this type of object.

SetLocation Method (Attribute Object)

Scope: Schematic editor

Object: Attribute Object

Prerequisites: None

Specifies the location for an attribute.

Note:
All coordinates are measured in 100ths of an inch.

Usage

Attribute.**SetLocation**(ByVal *X* As Long, ByVal *Y* As Long)

Arguments

- X

  The X coordinate for the attribute.
- Y

  The Y coordinate for the attribute.

SetObjectColor Method (Attribute Object)

Scope: Schematic editor

Object: Attribute Object

Prerequisites: None

Sets the color in which the Attribute is drawn.

Usage

Attribute.**SetObjectColor**(ByVal *newColor* as IColor)

Arguments

- newColor

  The new CColor Object assigned to the attribute.

Application Property (Attribute Object)

Scope: Schematic editor

Object: Attribute Object

Access: Read-Only

Prerequisites: None

Returns the Application Object.

Usage

Attribute.**Application**

Arguments

None

Return Values

IVdApp. The Application Object.

Description

See Application Object for more information.

Child Property (Attribute Object)

Scope: Schematic editor

Object: Attribute Object

Access: Read-Only

Prerequisites: None

Returns the child Block Object of the attribute.

Usage

Attribute.**Child**

Arguments

None

Return Values

Object. The child Block Object for this property.

Description

See Block Object for more information.

EitherValue Property (Attribute Object)

Scope: Schematic editor

Object: Attribute Object

Access: Read/Write

Prerequisites: None

Returns or sets the value of the attribute.

Usage

Attribute.**EitherValue** = String

Arguments

None

Return Values

String. A string that contains the attribute value.

Description

If the attribute has an instance (OAT) value, EitherValue applies to the instance (OAT) value (refer to "InstanceValue Property (Attribute Object)" on page 177), otherwise EitherValue applies to the regular value (refer to "Value Property (Attribute Object)" on page 186).

Font Property (Attribute Object)

Scope: Schematic editor

Object: Attribute Object

Access: Read/Write

Prerequisites: None

Returns or sets the font that is used to draw the attribute.

Usage

Attribute.**Font** = VdFont

Arguments

None

Return Values

VdFont. The return/set type for this property. This takes the form of VdFont Enum.

InstanceValue Property (Attribute Object)

Scope: Schematic editor

Object: Attribute Object

Access: Read/Write

Prerequisites: None

Returns or sets the instance (OAT) value for an attribute.

Usage

Attribute.**InstanceValue** = String

Arguments

None

Return Values

String. The string that specifies the value of this property.

If the attribute has no instance (OAT) value, the property value is an empty string.

Name Property (Attribute Object)

Scope: Schematic editor

Object: Attribute Object

Access: Read-Only

Prerequisites: None

Returns the name of the attribute.

Usage

Attribute.**Name**

Arguments

None

Return Values

String. The string that contains the name of this attribute in the form NAME=VALUE.

The Name property specifies the NAME portion, while the Value Property (Attribute Object) specifies the VALUE portion.

Examples

```
attr.Name="REFDES"
MsgBox attr.Name
```

Orientation Property (Attribute Object)

Scope: Schematic editor

Object: Attribute Object

Access: Read/Write

Prerequisites: None

Returns or sets the attribute orientation.

Usage

Attribute.**Orientation** = VdOrientation

Arguments

None

Return Values

VdOrientation. The return/set type for this property. This takes the form of VdOrientation Enum.

Origin Property (Attribute Object)

Scope: Schematic editor

Object: Attribute Object

Access: Read/Write

Prerequisites: None

Returns or sets one of the nine origins that is used for this attribute.

Usage

Attribute.**Origin** = VdOrigin

Arguments

None

Return Values

VdOrigin. The return/set type for this property. This is of the form VdOrigin Enum.

Parent Property (Attribute Object)

Scope: Schematic editor

Object: Attribute Object

Access: Read-Only

Prerequisites: None

Returns the parent Block Object for the attribute.

Usage

Attribute.**Parent**

Arguments

None

Return Values

Object. The parent Block Object for the attribute.

Selected Property (Attribute Object)

Scope: Schematic editor

Object: Attribute Object

Access: Write-Only

Prerequisites: None

Sets the selection status of the attribute.

Usage

Attribute.**Selected** = True | False

Arguments

None

Return Values

True | False. True - Selects the attribute. False - Deselects the attribute.

Size Property (Attribute Object)

Scope: Schematic editor

Object: Attribute Object

Access: Read/Write

Prerequisites: None

Returns or sets the text size for the attribute.

Usage

Attribute.**Size** = Long

Arguments

None

Return Values

Long. The value that specifies the text size for the attribute.

TextString Property (Attribute Object)

Scope: Schematic editor

Object: Attribute Object

Access: Read/Write

Prerequisites: None

Returns or sets the string associated with an attribute.

Usage

Attribute.**TextString** = String

Arguments

None

Return Values

String. A string that contains the value of the attribute.

This string is of the form NAME=VALUE. The Name Property (Attribute Object) specifies the NAME portion, while the Value Property (Attribute Object) specifies the VALUE portion. TextString specifies the full specification. If the attribute has an instance (OAT) value, setting or referencing this property affects that value.

Examples

```
MsgBox attr.TextString ' Prints NAME=VALUE
MsgBox attr.Name ' Prints NAME
MsgBox attr.Value ' Prints VALUE

' Set attribute's NAME to REFDES and VALUE (or instance value) to U7
attr.TextString="REFDES=U7"
```

Type Property (Attribute Object)

Scope: Schematic editor

Object: Attribute Object

Access: Read-Only

Prerequisites: None

Returns the attribute type.

Usage

Attribute.**Type**

Arguments

None

Return Values

VdObjectType. The return type for this property. This is in the form of VdObjectType Enum.

Value Property (Attribute Object)

Scope: Schematic editor

Object: Attribute Object

Access: Read/Write

Prerequisites: None

Returns or sets the attribute's regular value even if the attribute has an instance (OAT) value.

Usage

Attribute.**Value** = String

Arguments

None

Return Values

String. A string that contains the value of the attribute.

You can also use the EitherValue Property (Attribute Object) .

Examples

```
attr.Value="U3"
MsgBox attr.Value
```

Visible Property (Attribute Object)

Scope: Schematic editor

Object: Attribute Object

Access: Read/Write

Prerequisites: None

Returns or sets the visibility of the attribute.

Usage

Attribute.**Visible** = VdVisibilityFlag

Arguments

None

Return Values

VdVisibilityFlag. The return/set type for this property in the form VdVisibilityFlag Enum.

Block Object

This object represents a single sheet of a schematic or symbol.

The following table lists methods and properties of the Block object with links to the respective reference pages.

Table 8. Block Object Methods and Properties

Method or Property	Description
AddArc Method (Block Object)	Adds an arc to the block.
AddAttribute Method (Block Object)	Adds an unattached attribute to the block.
AddBatchAttributes Method (Block Object)	Adds multiple unattached attributes on the active block as an encoded string.
AddBox Method (Block Object)	Creates a Box Object within the block.
AddCircle Method (Block Object)	Adds a circle to the block.
AddFub Method (Block Object)	Adds a functional block (FUB) to the block.
AddLine Method (Block Object)	Adds a line to the block.
AddLine2 Method (Block Object)	Adds a line to the block.
AddNet Method (Block Object)	Adds a net to the block.
AddNetEx Method (Block Object)	Adds a net to the block.
AddPartInstance Method (Block Object)	Adds a part instance to the block.
AddPin Method (Block Object)	Adds a pin to a symbol block.
AddPinAtLocation Method (Block Object)	Adds a pin to a symbol block.
AddSymbolInstance Method (Block Object)	Adds a symbol instance to the block.
AddText Method (Block Object)	Adds annotation text string.
ApplySymbolUpdate Method (Block Object)	Updates components with changes from their symbols.
ChangeBorder Method (Block Object)	Changes from one border symbol to another on a schematic.
ChangeComponent Method (Block Object)	Exchanges one or more components with a different component.
ChangeComponentPreserveRefdes Method (Block Object)	Exchanges one or more components with a different component and preserves the REFDES.
ClearHighlight Method (Block Object)	Marks components as current and clears the highlight.
DeleteBorder Method (Block Object)	Deletes a border symbol from a schematic. The border need not be selected, although it can be.
DeleteSelected Method (Block Object)	Deletes all currently selected objects.

Table 8. Block Object Methods and Properties (continued)

Method or Property	Description
DeSelectAll Method (Block Object)	Clears selection of all objects on this block.
FindAttribute Method (Block Object)	Searches for an attribute by its name.
GetBatchAttributes Method (Block Object)	Returns the unattached attributes on the currently active block as an encoded string.
GetBboxPoint Method (Block Object)	Returns the location of a specified corner of a bounding box.
GetChildBlock Method (Block Object)	Allows access to the child block object.
GetName Method (Block Object)	Returns the hierarchical path to the active view in the current block.
InsertBorder Method (Block Object)	Inserts a border symbol onto a schematic.
PromoteSymbolNumbers Method (Block Object)	Updates components with pin number changes from their symbols.
RepositionAttributesAsOnSymbol Method (Block Object)	Updates components with position changes from the symbol.
SetZSheetSize Method (Block Object)	Specifies width and height for Z sized blocks.
UpdateBorder Method (Block Object)	Updates attribute values on a border symbol.
Application Property (Block Object)	Returns the Application Object.
Attributes Property (Block Object)	Returns a collection of Attribute Objects for the block.
DataType Property (Block Object)	Returns a value that relates whether the block is a schematic or a symbol.
IsFub Property (Block Object)	Returns a value that indicates whether the block is a functional block (FUB) or not.
LibraryName Property (Block Object)	Returns the library alias assigned to this block.
OpenMode Property (Block Object)	Returns a value that indicates how the Block was opened.
Parent Property (Block Object)	Returns the parent View Object of the block.
SheetNum Property (Block Object)	Returns the page number of the block.
SheetSize Property (Block Object)	Returns or sets the block drawing sheet size.
SymbolType Property (Block Object)	Returns or sets the symbol type used for the block.
Type Property (Block Object)	Returns the type for the block.
UpdateOutline Method (Block Object)	Updates the outline.

AddArc Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Adds an arc to the block.

Usage

Block.**AddArc**(ByVal *X1* As Long, ByVal *Y1* As Long, ByVal *X2* As Long, ByVal *Y2* As Long, ByVal *X3* As Long, ByVal *Y3* As Long) As IVdArc

Arguments

- X1

  X coordinate of the starting point.
- Y1

  Y coordinate of the starting point.
- X2

  X coordinate of any non ending point.
- Y2

  Y coordinate of any non ending point.
- X3

  X coordinate of the ending point.
- Y3

  Y coordinate of the ending point.

Return Values

As IVdArc. This is the newly-created Arc Object.

AddAttribute Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Adds an unattached attribute to the block.

Usage

Block.**AddAttribute**(ByVal *String* As String, ByVal *X* As Long, ByVal *Y* As Long, ByVal *Visibility* As VdVisibilityFlag) As IVdAttr

Arguments

- String

  String in the form NAME=VALUE.
- X

  The X coordinate of the attribute.
- Y

  The Y coordinate of the attribute.
- Visibility

  The visibility of the attribute in the form VdVisibilityFlag Enum.

Return Values

As IVdAttr. The newly added Attribute Object.

AddBatchAttributes Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Adds multiple unattached attributes on the active block as an encoded string.

Usage

Block.**AddBatchAttributes**(ByVal *AttributeListString* As String) As Boolean

Arguments

- AttributeListString

  Encoded string containing attributes to be added. The AttributeListString is specially formatted as shown below.

  ```
  <visibility> <duplicate> Name = Value<CR>
  (repeated for each attribute)
  ```

  <*visibility*> should be one of the following integers:

  - **0** — Invisible
  - **1** — Name and value visible
  - **2** — Name only visible
  - **3** — Value only visible

  <*duplicate*> should be one of the following integers:

  - **0** — add - add new attribute regardless of duplicates
  - **1** — replace - if attribute with this name exists, replace it

  The <CR> represents a carriage return character (ASCII 13).

Return Values

As Boolean:

- **True** — The attributes were added successfully.
- **False** — The attributes could not be added.

Examples

This example adds two attributes.

```
char* string="0 0 FOO=BAR\r1 0 TEST=BATCH"
pDisp->AddBatchAttributes(string);
```

AddBox Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Creates a Box Object within the block.

Usage

Block.**AddBox**(ByVal *LowerLeftx* As Long, ByVal *LowerLefty* As Long, ByVal *UpperRightx* As Long, ByVal *UpperRighty* As Long) As IVdBox

Arguments

- LowerLeftx

  X coordinate of the lower left hand corner of the box.
- LowerLefty

  Y coordinate of the lower left hand corner of the box.
- UpperRightx

  X coordinate of the upper right hand corner of the box.
- UpperRighty

  Y coordinate of the upper right hand corner of the box.

Return Values

As IVdBox. The newly added Box Object.

AddCircle Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Adds a circle to the block.

Usage

Block.**AddCircle**(ByVal *Centerx* As Long, ByVal *Centery* As Long, ByVal *Radius* As Long) As IVdCircle

Arguments

- Centerx

  X coordinate of the center of the circle.
- Centery

  Y coordinate of the center of the circle.
- Radius

  The length of the radius of the circle.

Return Values

As IVdCircle. The newly created Circle Object.

AddFub Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Adds a functional block (FUB) to the block.

Usage

Block.**AddFub**(ByVal *FubName* As String, ByVal *LowerLeftx* As Long, ByVal *LowerLefty* As Long, ByVal *UpperRightx* As Long, ByVal *UpperRighty* As Long) As IVdComp

Arguments

- FubName

  Name of the FUB to be instantiated. This string must be unique and not conflict with existing symbol names.
- LowerLeftx

  X coordinate of the lower left hand corner of the FUB.
- LowerLefty

  Y coordinate of the lower left hand corner of the FUB.
- UpperRightx

  X coordinate of the upper right hand corner of the FUB.
- UpperRighty

  Y coordinate of the upper right hand corner of the FUB.

Return Values

As IVdComp. The newly created FUB in the form of a Component Object.

AddLine Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Adds a line to the block.

Usage

Block.**AddLine**(ByVal *Pt1* As Long, ByVal *Pt2* As Long) As IVdLine

Arguments

- Pt1

  Starting coordinates (packed) of the line.
- Pt2

  Ending coordinates (packed) of the line.

Return Values

As IVdLine. The newly created Line Object.

Description

The two points passed in are packed into two longs, with low word representing the X coordinate and the high word representing the Y coordinate. For example:

```
MAKELONG(X,Y)
```

The X,Y coordinates are packed into a single long, in accordance with the C++ MAKELONG macro.

Examples

For Visual Basic, implement this function as follows:

```
Function LOWORD(ByVal dw)
 If dw And &H8000& Then
 LOWORD = dw Or &HFFFF0000
 Else
 LOWORD = dw And &HFFFF&
 End If
End Function

Public Function MAKELONG(LoByte, HiByte)
 If HiByte And &H80 Then
 MAKELONG = ((HiByte * &H100&) Or LoByte) Or &HFFFF0000
```

```
 Else
 MAKELONG = (HiByte * &H100) Or LoByte
 End If
End Function
```

AddLine2 Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Adds a line to the block.

Usage

Block.**AddLine2**(ByVal *X1* As Long, ByVal *Y1* As Long, ByVal *X2* As Long, ByVal *Y2* As Long) As IVdLine

Arguments

- X1

  X coordinate of the line's starting point.
- Y1

  Y coordinate of the line's starting point.
- X2

  X coordinate of the line's end point.
- Y2

  Y coordinate of the line's end point.

Return Values

As IVdLine. The newly created Line Object.

Description

The points passed in are packed into four longs, with X1 and Y1 representing the coordinates for the line's starting point and X2, Y2 representing the coordinates for the end point.

AddNet Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Adds a net to the block.

Usage

Block.**AddNet**(ByVal *Locationx1* As Long, ByVal *Locationy1* As Long, ByVal *Locationx2* As Long, ByVal *Locationy2* As Long, ByVal *CompPin1* As IVdCmpPin, ByVal *CompPin2* As IVdCmpPin, ByVal *BusOrWire* As VdBusOrWire) As IVdNet

Arguments

Note:
All coordinates are measured in 100ths of an inch.

- Locationx1

  Integer. The starting X location.
- Locationy1

  Integer. The starting Y location.
- Locationx2

  Integer. The ending X location.
- Locationy2

  Integer. The ending Y location.
- CompPin1

  IVdCmpPin. The first component pin. Ignored if set to NULL.
- CompPin2

  IVdCmpPin. The second component pin. Ignored if set to NULL.
- BusOrWire

  Adds a Bus or a Wire type net. Indicates what kind of net will be added (Bus) or (Wire). This is of the form VdBusOrWire Enum.

Return Values

As IVdNet. The newly created Net Object.

Description

The first coordinate pair (Locationx1, Locationy1) represents one end of the net; the second coordinate pair (Locationx2, Locationy2) represents the other end. Similarly, the first component pin (CompPin1) represents one end of the net, and the second component pin (CompPin2) represents the other end.

Component pin arguments take precedence over coordinate arguments, if both are provided.

You can add a net to a block between:

- two coordinate locations, x1,y1 and x2,y2
- two component pins, CompPin1 and CompPin2
- a combination of one location x,y and one component pin

Examples

The following example shows various combinations of the AddNet method:

```
Dim NullPin
 Set NullPin=Nothing ' Makes a NULL object

'Adds a wire net between two locations (100,100) and (0,0)
Set N1 = ActiveView.Block.AddNet(100,100,0,0,NullPin,NullPin,VD_WIRE)

'Adds a wire net between a location (100,100) and a component pin (Cp2)
Set N2 = ActiveView.Block.AddNet(100,100,0,0, NullPin, Cp2,VD_WIRE)

'Adds a wire net between two component pins (Cp1 and Cp2)
Set N3 = ActiveView.Block.AddNet(0,0,0,0,Cp1,Cp2,VD_WIRE)
```

AddNetEx Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Adds a net to the block.

Usage

Block.**AddNetEx**(ByVal *Locationx1* As Long, ByVal *Locationy1* As Long, ByVal *Locationx2* As Long, ByVal *Locationy2* As Long, ByVal *CompPin1* As IVdCmpPin, ByVal *CompPin2* As IVdCmpPin, ByVal *BusOrWire* As VdBusOrWire) As IVdNet

Arguments

Note:
All coordinates are measured in 100ths of an inch.

- Locationx1

  Integer. The starting X location.
- Locationy1

  Integer. The starting Y location.
- Locationx2

  Integer. The ending X location.
- Locationy2

  Integer. The ending Y location.
- CompPin1

  IVdCmpPin. The first component pin. Ignored if set to NULL.
- CompPin2

  IVdCmpPin. The second component pin. Ignored if set to NULL.
- BusOrWire

  Adds a Bus or a Wire type net. Indicates what kind of net will be added (Bus) or (Wire). This is of the form VdBusOrWire Enum.

Return Values

As IVdNet. The newly created Net Object.

Description

The first coordinate pair (Locationx1, Locationy1) represents one end of the net; the second coordinate pair (Locationx2, Locationy2) represents the other end. Similarly, the first component pin (CompPin1) represents one end of the net, and the second component pin (CompPin2) represents the other end.

Component pin arguments take precedence over coordinate arguments, if both are provided.

You can add a net to a block between:

- two coordinate locations, x1,y1 and x2,y2
- two component pins, CompPin1 and CompPin2
- a combination of one location x,y and one component pin

Examples

The following example shows various combinations of the AddNetEx method:

```
Dim NullPin
 Set NullPin=Nothing ' Makes a NULL object

'Adds a wire net between two locations (100,100) and (0,0)
Set N1 = ActiveView.Block.AddNet(100,100,0,0,NullPin,NullPin,VD_WIRE)

'Adds a wire net between a location (100,100) and a component pin (Cp2)
Set N2 = ActiveView.Block.AddNet(100,100,0,0, NullPin, Cp2,VD_WIRE)

'Adds a wire net between two component pins (Cp1 and Cp2)
Set N3 = ActiveView.Block.AddNet(0,0,0,0,Cp1,Cp2,VD_WIRE)
```

AddPartInstance Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Adds a part instance to the block.

Usage

Block.**AddPartInstance** (ByVal *SymbolPartitionName* As String, ByVal *DeviceName* As String, ByVal *SymbolName* As String, ByVal *LocationX* As Long, ByVal *LocationY* As Long) As IVdComp

Arguments

Note:
All coordinates are measured in 100ths of an inch.

- SymbolPartitionName

  Name of the partition.
- DeviceName

  Name of the device.
- SymbolName

  Name of the symbol to be instantiated.
- LocationX

  X coordinate of the instance.
- LocationY

  Y coordinate of the instance.

Return Values

As IVdComp. Added part instance, in the form of a Component Object.

Examples

```
Set NewDevice_1 =
 myBlock.AddPartInstance("MISC","PCI-CONN","CON1L",200,200)
```

AddPin Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Adds a pin to a symbol block.

Usage

Block.**AddPin**(ByVal *X* As Long, ByVal *Y* As Long) As IVdPin

Arguments

Note:
All coordinates are measured in 100ths of an inch.

- X

  X Coordinate of the interior location.
- Y

  Y Coordinate of the interior location.

Return Values

As IVdPin. The newly created Pin Object.

Description

The block where you add a pin must be of type symbol. The pin will start at the internal point specified and the other end of the pin will snap to the closest edge of the symbol bounding box.

AddPinAtLocation Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Adds a pin to a symbol block.

Note:
Since the VX.2.1 release, this method adds a default label to the pin.

Usage

Block.**AddPinAtLocation**(ByVal *X1* As Long, ByVal *Y1* As Long, ByVal *X2* As Long, ByVal *Y2* As Long) As IVdPin

Arguments

Note:
All coordinates are measured in 100ths of an inch.

- X1

  X coordinate of the first point.
- Y1

  Y coordinate of the first point.
- X2

  X coordinate of the second point.
- Y2

  X coordinate of the second point.

Return Values

As IVdPin. The newly created Pin Object.

Description

The block must be a symbol. The pin will start and finish at the internal points specified.

AddSymbolInstance Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Adds a symbol instance to the block.

Usage

Block.**AddSymbolInstance** (ByVal *SymbolPartitionName* As String, ByVal *SymbolName* As String, ByVal *Locationx* As Long, ByVal *Locationy* As Long) As IVdComp

Arguments

Note:
All coordinates are measured in 100ths of an inch.

- SymbolPartitionName

  Name of the partition.
- SymbolName

  Name of the symbol to be instantiated.
- Locationx

  X coordinate of the instance.
- Locationy

  Y coordinate of the instance.

Return Values

As IVdComp. The newly created and added symbol instance, returned as a Component Object.

AddText Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Adds annotation text string.

Usage

Block.**AddText**(ByVal *TextString* As String, ByVal *X* As Long, ByVal *Y* As Long) As IVdText

Arguments

- TextString

  String to be added.
- X

  X coordinate.
- Y

  Y coordinate.

Return Values

As IVdText. The newly created Text Object.

Description

The string will be located at the coordinates passed. To adjust the orientation or other properties of the Text string, use the returned Text object and its properties and methods to make the adjustments.

Examples

Add a red annotation string.

```
Set Txt=ActiveView.Block.AddText("FOOBAR", 100, 100)
Txt.Color = VGCOLORRED
Txt.Size = 20
```

ApplySymbolUpdate Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Updates components with changes from their symbols.

Usage

Block.**ApplySymbolUpdate**(ByVal *SelectedOnly* As VdAllOrSelected, ByVal *Slot* As Long)

Arguments

- SelectedOnly

  Specifies whether to consider all objects of just the selected objects. This is of the form VdAllOrSelected Enum.

- Slot

  Slot value to use when promoting symbol pin numbers. Use 1 as the value for non-slotted parts.

Description

The method updates the position of symbol pin numbers and the REFDES that were moved after the component was instantiated. The method also corrects the pin number values based on the slot specified in the second parameter.

ChangeBorder Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Changes from one border symbol to another on a schematic.

Usage

Block.**ChangeBorder**(ByVal *NewBorder* As String)

Arguments

- NewBorder

  Name of new border symbol that will replace existing border.

ChangeComponent Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Exchanges one or more components with a different component.

Usage

Block.**ChangeComponent**(ByVal *OldComp* As String, ByVal *NewComp* As String) As Boolean

Arguments

- OldComp

  Name of the component to be replaced. The special string "< *Selected Components* >" will cause all the components on the selected list to be replaced by the new component.

- NewComp

  Name of the component which replaces OldComp.

Description

The special string "< *Selected Components* >" causes all the components on the selected list to be replaced by the new component.

This method returns a Boolean True if the component was replaced, or False if the component could not be replaced. False throws an Automation exception if the special string is used and there are not any components on the selected list.

Examples

Replace all components with a resistor.

```
ActiveView.Block.ChangeComponent "< Selected Components >", "resistor.1"
'Replace all capacitors with resistors
ActiveView.Block.ChangeComponent "cap.1", "resistor.1"
```

ChangeComponentPreserveRefdes Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Exchanges one or more components with a different component and preserves the REFDES.

Usage

Block.**ChangeComponentPreserveRefdes**(ByVal *OldComp* As String, ByVal *NewComp* As String) As Boolean

Arguments

- OldComp

  Name of the component to be replaced. The special string "< *Selected Components* >" will cause all the components on the selected list to be replaced by the new component.

- NewComp

  Name of the component which replaces OldComp.

Description

The special string "< *Selected Components* >" causes all the components on the selected list to be replaced by the new component.

This method returns True if the component was replaced, or False if the component could not be replaced. False throws an Automation exception if the special string is used and there are not any components on the selected list.

Examples

Replace all components while preserving the REFDES.

```
ActiveView.Block.ChangeComponentPreserveRefdes "< Selected Components >",
 "0805cap.1"
'Replace all capacitors with surface mount version
ActiveView.Block.ChangeComponent "cap.1", "0805cap.1"
```

ClearHighlight Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Marks components as current and clears the highlight.

Note:
This method should not be called until all desired changes have been updated. It will disable all further updates using VD_ALL since there will be no out-of-date components left. Updates using VD_SELECTED can still be made on components that are selected.

Usage

Block.**ClearHighlight**(ByVal *SelectedOnly* As VdAllOrSelected)

Arguments

- SelectedOnly

  Specifies whether to consider all objects or just the selected ones. This is of the form VdAllOrSelected Enum.

Description

This method updates the component version date-time-stamp to agree with its symbol. In addition, the out-of-date flag is cleared and the out-of-date highlight removed.

DeleteBorder Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Deletes a border symbol from a schematic. The border need not be selected, although it can be.

Usage

Block.**DeleteBorder**()

Arguments

None

DeleteSelected Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Deletes all currently selected objects.

Usage

Block.**DeleteSelected**(ByVal *delUnc* As Boolean)

Arguments

- delUnc

  Controls what happens to unselected nets.

  True - All selected items (including dangling nets) will be deleted.

  False - All selected items (except dangling nets) will be deleted.

DeSelectAll Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Clears selection of all objects on this block.

Usage

Block.**DeSelectAll**()

Arguments

None

FindAttribute Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Searches for an attribute by its name.

Usage

Block.**FindAttribute**(ByVal *AttributeName* As String) As IVdAttr

Arguments

- AttributeName

  The name of the attribute that is the object of the search.

Return Values

As IVdAttr. This is the Attribute Object if found, otherwise it is null.

Examples

This example finds the PageNum attribute.

```
Set PageNumAttr=ActiveView.Block.FindAttribute("PAGENUM")
If PageNumAttr Is Nothing Then
Else
   PageNumAttr.Value="2"
End If
```

GetBatchAttributes Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Returns the unattached attributes on the currently active block as an encoded string.

Usage

Block.**GetBatchAttributes**() As String

Arguments

None

Return Values

As String. The string of attributes, specially formatted as follows:

```
Visibility Name=Value<CR>
(repeated for each attribute)
```

Where *Visibility* is an integer represented by VdVisibilityFlag Enum and the <CR> here represents a carriage return character (ASCII 13).

GetBboxPoint Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Returns the location of a specified corner of a bounding box.

Usage

Block.**GetBboxPoint**(ByVal *Location* As VdCorner) As IVdPoint

Arguments

Note:
All coordinates are measured in 100ths of an inch.

- Location

  Specifies which VdCorner Enum location (LowerLeft or UpperRight).

Return Values

As IVdPoint. This is a Point Object containing coordinates.

GetChildBlock Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Allows access to the child block object.

Note:
Use this method for composite symbols only.

Usage

Block.**GetChildBlock**() As IVdBlock

Arguments

None

Return Values

IVdBlock. The child Block Object.

Examples

Use this method to provide access to a schematic sheet block for a composite symbol.

GetName Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Returns the hierarchical path to the active view in the current block.

Usage

Block.**GetName**(ByVal *Flag* As VdNameType) As String

Arguments

- Flag

  Specifies the nature of the path that is returned by this method:

  FULL_PATH_NAME — Returns full hierarchical path specification. The path you specify can use either component names (labels) or UIDs. For example, top_block\block1\block2.

  SHORT_NAME — Name without the hierarchical path.

  This is of the form VdNameType Enum.

Return Values

As String. A string containing the hierarchical path.

InsertBorder Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Inserts a border symbol onto a schematic.

Usage

Block.**InsertBorder**()

Arguments

None

The default border symbol is set in project settings for the specified sheet size.

PromoteSymbolNumbers Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Updates components with pin number changes from their symbols.

Usage

Block.**PromoteSymbolNumbers**(ByVal *SelectedOnly* As VdAllOrSelected, ByVal *Slot* As Long)

Arguments

- SelectedOnly

  Specifies whether to consider all objects or just the selected items. This is of the form VdAllOrSelected Enum.

- Slot

  Slot value to use when promoting symbol pin numbers. Use 1 as the value for non-slotted parts.

Description

This method corrects the pin number values with pin number changes from their symbols based on the slot specified in the second parameter.

RepositionAttributesAsOnSymbol Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Updates components with position changes from the symbol.

Usage

Block.**RepositionAttributesAsOnSymbol**(ByVal *SelectedOnly* As VdAllOrSelected)

Arguments

- SelectedOnly

  Specifies whether to consider all objects or just the selected items. This is of the form VdAllOrSelected Enum.

Description

This method updates the position of symbol pin numbers and the REFDES that were moved after the component was instantiated.

SetZSheetSize Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Specifies width and height for Z sized blocks.

Usage

Block.**SetZSheetSize**(ByVal *Width* As Long, ByVal *Height* As Long)

Arguments

- Width

  Width of the block (expressed in tenths of an inch).
- Height

  Height of the block (expressed in tenths of an inch).

UpdateBorder Method (Block Object)

Scope: Schematic editor

Object: Block Object

Prerequisites: None

Updates attribute values on a border symbol.

Usage

Block.**UpdateBorder**()

Arguments

None

Application Property (Block Object)

Scope: Schematic editor

Object: Block Object

Access: Read-Only

Prerequisites: None

Returns the Application Object.

Usage

Block.**Application**

Arguments

None

Return Values

IVdApp. The return type for this property.

Description

See Application Object for more information.

Attributes Property (Block Object)

Scope: Schematic editor

Object: Block Object

Access: Read-Only

Prerequisites: None

Returns a collection of Attribute Objects for the block.

Usage

Block.**Attributes**

Arguments

None

Return Values

IVdObjs. The Attribute Objects for the block.

Description

See Attribute Object for more information.

DataType Property (Block Object)

Scope: Schematic editor

Object: Block Object

Access: Read-Only

Prerequisites: None

Returns a value that relates whether the block is a schematic or a symbol.

Usage

Block.**DataType**

Arguments

None

Return Values

VdDataType. The return type for this property. This is of the form VdDataType Enum.

IsFub Property (Block Object)

Scope: Schematic editor

Object: Block Object

Access: Read-Only

Prerequisites: None

Returns a value that indicates whether the block is a functional block (FUB) or not.

Usage

Block.**IsFub**

Arguments

None

Return Values

True | False. True - the block is a FUB. False - the block is not a FUB.

LibraryName Property (Block Object)

Scope: Schematic editor

Object: Block Object

Access: Read-Only

Prerequisites: None

Returns the library alias assigned to this block.

Usage

Block.**LibraryName**

Arguments

None

Return Values

String. A string containing the library alias assigned to the block. An empty string “” can return if the block was loaded by using the search order mechanism instead of a library alias.

OpenMode Property (Block Object)

Scope: Schematic editor

Object: Block Object

Access: Read-Only

Prerequisites: None

Returns a value that indicates how the Block was opened.

Usage

Block.**OpenMode**

Arguments

None

Return Values

VdOpenMode. The return type for this property. This is of the form VdOpenMode Enum.

Parent Property (Block Object)

Scope: Schematic editor

Object: Block Object

Access: Read-Only

Prerequisites: None

Returns the parent View Object of the block.

Usage

Block.**Parent**

Arguments

None

Return Values

IVdView. The parent View Object of the block.

SheetNum Property (Block Object)

Scope: Schematic editor

Object: Block Object

Access: Read-Only

Prerequisites: None

Returns the page number of the block.

Usage

Block.**SheetNum**

Arguments

None

Return Values

String. A string containing the page number of this block.

SheetSize Property (Block Object)

Scope: Schematic editor

Object: Block Object

Access: Read/Write

Prerequisites: None

Returns or sets the block drawing sheet size.

Usage

Block.**SheetSize** = VdSheetSize

Arguments

None

Return Values

VdSheetSize. The return/set type for this property. This is of the form VdSheetSize Enum.

SymbolType Property (Block Object)

Scope: Schematic editor

Object: Block Object

Access: Read/Write

Prerequisites: None

Returns or sets the symbol type used for the block.

Note:
This property only applies to symbol blocks.

Usage

Block.**SymbolType** = VdSymbolType

Arguments

None

Return Values

VdSymbolType. The return/set type for this property. This is of the form VdSymbolType Enum.

Type Property (Block Object)

Scope: Schematic editor

Object: Block Object

Access: Read-Only

Prerequisites: None

Returns the type for the block.

Usage

Block.**Type**

Arguments

None

Return Values

VdObjectType. The return type for this property. This is of the form VdObjectType Enum.

UpdateOutline Method (Block Object)

Prerequisites: None

Object: Block Object

Updates the outline.

Usage

```
Block.UpdateOutline() As Boolean
```

Arguments

None

Return Values

Boolean.

Box Object

The Box is a graphical object in Xpedition Designer.

The following table lists methods and properties of the Box object with links to the respective reference pages.

Table 9. Box Object Methods and Properties

Method or Property	Description
GetLocation Method (Box Object)	Returns lower left and upper right coordinates that specify the location of a box.
GetObjectColor Method (Box Object)	Gets the color in which the box is drawn.
GetObjectFillColor Method (Box Object)	Gets the fill color for the box.
IsColorAutomatic Method (Box Object)	Determines if the box has been assigned an automatic color.
IsFillColorAutomatic Method (Box Object)	Determines if the box has been assigned an automatic fill color.
SetAutomaticColor Method (Box Object)	Sets or unsets the color of a box object as the automatic color for boxes.
SetAutomaticFillColor Method (Box Object)	Sets or unsets the fill color of a box object as the automatic fill color for boxes.
SetLocation Method (Box Object)	Specifies the box location using two points (X1,Y1) and (X2,Y2).
SetObjectColor Method (Box Object)	Sets the color in which the box is drawn.
SetObjectFillColor Method (Box Object)	Sets the fill color for the box.
Application Property (Box Object)	Returns the Application Object.
FillStyle Property (Box Object)	Returns or sets the fill style in which the box is drawn.
LineStyle Property (Box Object)	Returns or sets the line style in which the box is drawn.
Parent Property (Box Object)	Returns the parent Block Object for the box.
Selected Property (Box Object)	Sets the selection status for the box.
Type Property (Box Object)	Returns the type for the box.

GetLocation Method (Box Object)

Scope: Schematic editor

Object: Box Object

Prerequisites: None

Returns lower left and upper right coordinates that specify the location of a box.

Usage

Box.**GetLocation**(ByVal *Flag* As VdCorner) As IVdPoint

Arguments

Note:
All coordinates are measured in 100ths of an inch.

- Flag

  Specified point location to be returned. This is of the form VdCorner Enum.

Return Values

As IVdPoint. Point Object containing coordinates:

- VdCorner.LowerLeft
- VdCorner.UpperRight

GetObjectColor Method (Box Object)

Scope: Schematic editor

Object: Box Object

Prerequisites: None

Gets the color in which the box is drawn.

Usage

Box.**GetObjectColor**() As IColor

Arguments

None

Return Values

As IColor. The CColor Object of the box.

GetObjectFillColor Method (Box Object)

Scope: Schematic editor

Object: Box Object

Prerequisites: None

Gets the fill color for the box.

Usage

Box.**GetObjectFillColor**() As IColor

Arguments

None

Return Values

As IColor. The CColor Object of the box.

IsColorAutomatic Method (Box Object)

Scope: Schematic editor

Object: Box Object

Prerequisites: None

Determines if the box has been assigned an automatic color.

Usage

Box.**IsColorAutomatic**() As Boolean

Arguments

None

Return Values

As Boolean. True - there is an automatic color set for the attribute. False - no automatic color is set for the attribute.

For more information, see the "SetAutomaticColor Method (Box Object)" on page 245.

IsFillColorAutomatic Method (Box Object)

Scope: Schematic editor

Object: Box Object

Prerequisites: None

Determines if the box has been assigned an automatic fill color.

Usage

Box.**IsFillColorAutomatic**() As Boolean

Arguments

None

Return Values

As Boolean. True - there is an automatic fill color set for the box. False - no automatic color is set for the box.

For more information, see the "SetAutomaticFillColor Method (Box Object)" on page 246.

SetAutomaticColor Method (Box Object)

Scope: Schematic editor

Object: Box Object

Prerequisites: None

Sets or unsets the color of a box object as the automatic color for boxes.

Usage

Box.**SetAutomaticColor**(byVal *bAutomatic* as Boolean)

Arguments

- bAutomatic

  True - sets the automatic color for the box.

  False - unsets the automatic color for the box.

  If an object has automatic color, as determined by the IsColorAutomatic Method (Box Object), the actual color of the object is the default for this type of object.

SetAutomaticFillColor Method (Box Object)

Scope: Schematic editor

Object: Box Object

Prerequisites: None

Sets or unsets the fill color of a box object as the automatic fill color for boxes.

Usage

Box.**SetAutomaticFillColor**(byVal *bAutomatic* as Boolean)

Arguments

- bAutomatic

  True - sets the automatic fill color for the box.

  False - unsets the automatic fill color for the box.

If an object has automatic fill color, as determined by the "IsFillColorAutomatic Method (Box Object)" on page 244, the actual fill color of the object is the default for this type of object.

SetLocation Method (Box Object)

Scope: Schematic editor

Object: Box Object

Prerequisites: None

Specifies the box location using two points (X1,Y1) and (X2,Y2).

Usage

Box.**SetLocation**(ByVal *X1* As Long, ByVal *Y1* As Long, ByVal *X2* As Long, ByVal *Y2* As Long)

Arguments

Note:
All coordinates are measured in 100ths of an inch.

- X1

  X coordinate of the lower left hand corner.
- Y1

  Y coordinate of the lower left hand corner.
- X2

  X coordinate of the upper right hand corner.
- Y2

  Y coordinate of the upper right hand corner.

SetObjectColor Method (Box Object)

Scope: Schematic editor

Object: Box Object

Prerequisites: None

Sets the color in which the box is drawn.

Usage

Box.**SetObjectColor**(ByVal *newColor* as IColor)

Arguments

- newColor

  The new CColor Object of the box.

SetObjectFillColor Method (Box Object)

Scope: Schematic editor

Object: Box Object

Prerequisites: None

Sets the fill color for the box.

Usage

Box.**SetObjectFillColor**(ByVal *newColor* as IColor)

Arguments

- newColor

  The new CColor Object of the box.

Application Property (Box Object)

Scope: Schematic editor

Object: Box Object

Access: Read-Only

Prerequisites: None

Returns the Application Object.

Usage

Box.**Application**

Arguments

None

Return Values

IVdApp. The Application Object for the box.

Description

See Application Object for more information.

FillStyle Property (Box Object)

Scope: Schematic editor

Object: Box Object

Access: Read/Write

Prerequisites: None

Returns or sets the fill style in which the box is drawn.

Usage

Box.**FillStyle** = VdFillStyle

Arguments

None

Return Values

VdFillStyle. The return/set type for this property. This is of the form VdFillStyle Enum.

LineStyle Property (Box Object)

Scope: Schematic editor

Object: Box Object

Access: Read/Write

Prerequisites: None

Returns or sets the line style in which the box is drawn.

Usage

Box.**LineStyle** = VdLineStyle

Arguments

None

Return Values

VdLineStyle. The return/set type for this property. This is of the form VdLineStyle Enum.

Parent Property (Box Object)

Scope: Schematic editor

Object: Box Object

Access: Read-Only

Prerequisites: None

Returns the parent Block Object for the box.

Usage

Box.**Parent**

Arguments

None

Return Values

IVdBlock. The parent Block Object for the box.

Selected Property (Box Object)

Scope: Schematic editor

Object: Box Object

Access: Write-Only

Prerequisites: None

Sets the selection status for the box.

Usage

Box.**Selected** = True | False

Arguments

None

Return Values

True | False. True - the box is selected. False - the box is deselected.

Type Property (Box Object)

Scope: Schematic editor

Object: Box Object

Access: Read-Only

Prerequisites: None

Returns the type for the box.

Usage

Box.**Type**

Arguments

None

Return Values

VdObjectType. The return type for this property. This is of the form VdObjectType Enum.

CColor Object

The CColor object represents a color in RGB format.

The following table lists properties of the CColor object with links to the respective reference pages.

Table 10. CColor Object Properties

Property	Description
b Property (CColor Object)	Returns or sets the B (blue) component of color in RGB format.
g Property (CColor Object)	Returns or sets the G (green) component of color in RGB format. The value for green is 94.
r Property (CColor Object)	Returns or sets the R (red) component of color in RGB format.

b Property (CColor Object)

Scope: Schematic editor

Object: CColor Object

Access: Read/Write

Prerequisites: None

Returns or sets the B (blue) component of color in RGB format.

Usage

CColor.**b** = Long

Arguments

None

Return Values

Long. A value of 236 (the value for blue).

g Property (CColor Object)

Scope: Schematic editor

Object: CColor Object

Access: Read/Write

Prerequisites: None

Prerequisites: None.

Returns or sets the G (green) component of color in RGB format. The value for green is 94.

Usage

CColor.**g** = Long

Arguments

None

Return Values

Long. A value of 94.

r Property (CColor Object)

Scope: Schematic editor

Object: CColor Object

Access: Read/Write

Prerequisites: None

Returns or sets the R (red) component of color in RGB format.

Usage

CColor.**r** = Long

Arguments

None

Return Values

Long. A value of 128 for red.

Circle Object

This object represents a graphical circle on a schematic.

The following tables list methods and properties of the Circle object with links to the respective reference pages:

Table 11. Circle Object Methods and Properties

Method or Property	Description
GetCenter Method (Circle Object)	Returns the coordinates of the center of the circle.
GetObjectColor Method (Circle Object)	Gets the color in which the circle is drawn.
GetObjectFillColor Method (Circle Object)	Gets the fill color for the circle.
IsColorAutomatic Method (Circle Object)	Determines if the circle has been assigned an automatic color.
IsFillColorAutomatic Method (Circle Object)	Determines if the circle has been assigned an automatic fill color.
SetAutomaticColor Method (Circle Object)	Sets or unsets the color of a circle object as the automatic color for circles.
SetAutomaticFillColor Method (Circle Object)	Sets or unsets the fill color of a circle object as the automatic fill color for circles.
SetCenter Method (Circle Object)	Sets the coordinates of the center of a circle.
SetObjectColor Method (Circle Object)	Sets the color in which the circle is drawn.
SetObjectFillColor Method (Circle Object)	Sets the fill color for the circle.
Application Property (Circle Object)	Returns the Application Object of the circle.
FillStyle Property (Circle Object)	Returns or sets the fill style in which the circle is drawn.
LineStyle Property (Circle Object)	Returns or sets the line style in which the circle is drawn.
Parent Property (Circle Object)	Returns the parent Block Object for the circle.
Radius Property (Circle Object)	Returns or sets the radius for the circle.
Selected Property (Circle Object)	Sets the selection status of the circle.
Type Property (Circle Object)	Returns the type for the circle.

GetCenter Method (Circle Object)

Scope: Schematic editor

Object: Circle Object

Prerequisites: None

Returns the coordinates of the center of the circle.

Usage

Circle.**GetCenter**() As IVdPoint

Arguments

None

Return Values

As IVdPoint. The Point Object containing center coordinates.

GetObjectColor Method (Circle Object)

Scope: Schematic editor

Object: Circle Object

Prerequisites: None

Gets the color in which the circle is drawn.

Usage

Circle.**GetObjectColor**() As IColor

Arguments

None

Return Values

As IColor. The CColor Object representing the current color of the circle.

GetObjectFillColor Method (Circle Object)

Scope: Schematic editor

Object: Circle Object

Prerequisites: None

Gets the fill color for the circle.

Usage

Circle.**GetObjectFillColor**() As IColor

Arguments

None

Return Values

As IColor. The CColor Object of the circle.

IsColorAutomatic Method (Circle Object)

Scope: Schematic editor

Object: Circle Object

Prerequisites: None

Determines if the circle has been assigned an automatic color.

Usage

Circle.**IsColorAutomatic**() As Boolean

Arguments

None

Return Values

As Boolean. True - there is an automatic color set for the circle. False - no automatic color is set for the circle.

For more information, see the "SetAutomaticColor Method (Circle Object)" on page 266.

IsFillColorAutomatic Method (Circle Object)

Scope: Schematic editor

Object: Circle Object

Prerequisites: None

Determines if the circle has been assigned an automatic fill color.

Usage

Circle.**IsFillColorAutomatic**() As Boolean

Arguments

None

Return Values

As Boolean. True - there is an automatic fill color set for the circle. False - no automatic color is set for the circle.

For more information, see the "SetAutomaticFillColor Method (Circle Object)" on page 267.

SetAutomaticColor Method (Circle Object)

Scope: Schematic editor

Object: Circle Object

Prerequisites: None

Sets or unsets the color of a circle object as the automatic color for circles.

Usage

Circle.**SetAutomaticColor**(byVal *bAutomatic* as Boolean)

Arguments

- bAutomatic.

  The color that is set as the default for the object. True - sets automatic color for the circle. False - unsets automatic color.

  If an object has automatic color, as determined by the IsColorAutomatic Method (Circle Object), the actual color of the object is the default for this type of object.

SetAutomaticFillColor Method (Circle Object)

Scope: Schematic editor

Object: Circle Object

Prerequisites: None

Sets or unsets the fill color of a circle object as the automatic fill color for circles.

Usage

Circle.**SetAutomaticFillColor**(byVal *bAutomatic* as Boolean)

Arguments

- bAutomatic

  True - sets the automatic fill color for the circle.

  False - unsets the automatic fill color for the circle.

If an object has automatic fill color, as determined by the "IsFillColorAutomatic Method (Circle Object)" on page 265, the actual fill color of the object is the default for this type of object.

SetCenter Method (Circle Object)

Scope: Schematic editor

Object: Circle Object

Prerequisites: None

Sets the coordinates of the center of a circle.

Usage

Circle.**SetCenter**(ByVal *X* As Long, ByVal *Y* As Long)

Arguments

- X

  X coordinate of the center.
- Y

  Y coordinate of the center.

SetObjectColor Method (Circle Object)

Scope: Schematic editor

Object: Circle Object

Prerequisites: None

Sets the color in which the circle is drawn.

Usage

Circle.**SetObjectColor**(ByVal *newColor* as IColor)

Arguments

- newColor

  The new color assigned to the circle, as a CColor Object.

SetObjectFillColor Method (Circle Object)

Scope: Schematic editor

Object: Circle Object

Prerequisites: None

Sets the fill color for the circle.

Usage

Circle.**SetObjectFillColor**(ByVal *newColor* as IColor)

Arguments

- newColor

  The new CColor Object of the circle.

Application Property (Circle Object)

Scope: Schematic editor

Object: Circle Object

Access: Read-Only

Prerequisites: None

Returns the Application Object of the circle.

Usage

Circle.**Application**

Arguments

None

Return Values

IVdApp. The Application Object.

Description

See Application Object for more information.

FillStyle Property (Circle Object)

Scope: Schematic editor

Object: Circle Object

Access: Read/Write

Prerequisites: None

Returns or sets the fill style in which the circle is drawn.

Usage

Circle.**FillStyle** = VdFillStyle

Arguments

None

Return Values

VdFillStyle. The return/set type for this property. This is of the form VdFillStyle Enum.

LineStyle Property (Circle Object)

Scope: Schematic editor

Object: Circle Object

Access: Read/Write

Prerequisites: None

Returns or sets the line style in which the circle is drawn.

Usage

Circle.**LineStyle** = VdLineStyle

Arguments

None

Return Values

VdLineStyle. The return/set type for this property. This is of the form of VdLineStyle Enum.

Parent Property (Circle Object)

Scope: Schematic editor

Object: Circle Object

Access: Read-Only

Prerequisites: None

Returns the parent Block Object for the circle.

Usage

Circle.**Parent**

Arguments

None

Return Values

IVdBlock. The parent Block Object for the circle.

Radius Property (Circle Object)

Scope: Schematic editor

Object: Circle Object

Access: Read/Write

Prerequisites: None

Returns or sets the radius for the circle.

Usage

Circle.**Radius** = Long

Arguments

None

Return Values

Long. A value representing the radius of the circle.

Selected Property (Circle Object)

Scope: Schematic editor

Object: Circle Object

Access: Write-Only

Prerequisites: None

Sets the selection status of the circle.

Usage

Circle.**Selected** = True | False

Arguments

None

Return Values

True | False. The set type for this property. True - the circle is selected. False - the circle is deselected.

Type Property (Circle Object)

Scope: Schematic editor

Object: Circle Object

Access: Read-Only

Prerequisites: None

Returns the type for the circle.

Usage

Circle.**Type**

Arguments

None

Return Values

VdObjectType. The return type for this property. This is of the form VdObjectType Enum.

CommandsManager Object

The CommandsManager object allows the user to perform command-related actions: (un)register, execute, enable/disable commands).

The following table lists methods of the CommandsManager object with links to the respective reference pages.

Table 12. CommandsManager Object Methods

Method	Description
CommandDisable Method (CommandsManager Object)	Temporarily disables a command line command.
CommandEnable Method (CommandsManager Object)	Enables a command line command.
CommandRemove Method (CommandsManager Object)	Removes a command line command from the current session.
ExecuteCommand Method (CommandsManager Object)	Executes a command line command.
ExecuteMenuCommand Method (CommandsManager Object)	Causes the specified menu command to be executed.
RegisterOLECommand Method (CommandsManager Object)	Defines a new command line command.
UnregisterOLECommand Method (CommandsManager Object)	Revokes a previously registered command.

CommandDisable Method (CommandsManager Object)

Scope: Schematic editor

Object: CommandsManager Object

Prerequisites: None

Temporarily disables a command line command.

Usage

CommandsManager.**CommandDisable**(ByVal *CommandName* As String) As Boolean

Arguments

- CommandName

  The name of the command to disable.

Return Values

As Boolean. True - the command has been disabled (success). False - the command could not be disabled (failure).

Description

The command remains disabled until it is named as the argument for the CommandEnable Method (CommandsManager Object).

CommandEnable Method (CommandsManager Object)

Scope: Schematic editor

Object: CommandsManager Object

Prerequisites: None

Enables a command line command.

Usage

CommandsManager.**CommandEnable**(ByVal *CommandName* As String) As Boolean

Arguments

- CommandName

  The name of the command to enable.

Return Values

As Boolean. True - the command has been enabled (success). False - the command could not be enabled (failure).

Description

This method also works to enable commands that have been disabled by the CommandDisable Method (CommandsManager Object).

CommandRemove Method (CommandsManager Object)

Scope: Schematic editor

Object: CommandsManager Object

Prerequisites: None

Removes a command line command from the current session.

Usage

CommandsManager.**CommandRemove**(ByVal *CommandName* As String) As Boolean

Arguments

- CommandName

  The name of the command to remove.

Return Values

As Boolean. True - the command has been removed (success). False - the command could not be removed (failure).

ExecuteCommand Method (CommandsManager Object)

Scope: Schematic editor

Object: CommandsManager Object

Prerequisites: None

Executes a command line command.

Usage

CommandsManager.**ExecuteCommand**(ByVal *CommandString* As String) As Boolean

Arguments

- CommandString

  The command to be executed, including all arguments and other required data.

Return Values

As Boolean. True - The command was executed (success). False - The command could not be executed (failure).

Description

The Xpedition Designer command line provides a mechanism for you to specify commands by providing a string that represents a recognized Xpedition Designer command. This string must fully specify the command and provide all required data for that command.

Examples

In this example, the method invokes the “zselect” command.

```
ExecuteCommand "zselect"
```

ExecuteMenuCommand Method (CommandsManager Object)

Scope: Schematic editor

Object: CommandsManager Object

Prerequisites: None

Causes the specified menu command to be executed.

Usage

CommandsManager.**ExecuteMenuCommand**(ByVal *command_name* As String)

Arguments

- command_name

  Specifies the name of the menu command. The name is the same name used for the KeyBindings. For example to execute a **File > Open > Project** menu command, pass "FileOpenProject" as the string.

RegisterOLECommand Method (CommandsManager Object)

Scope: Schematic editor

Object: CommandsManager Object

Prerequisites: None

Defines a new command line command.

Usage

CommandsManager.**RegisterOLECommand**(ByVal *CommandName* As String, ByVal *CommandDescription* As String, ByVal *bModifiesData* As Boolean, ByVal *pDispatch* As Object) As Boolean

Arguments

- CommandName

  This is the unique name for the command.

- CommandDescription

  This is a description of the command function that will appear when there is a help query.

- bModifiesData

  A value of True indicates that this command modifies data. This causes the application to lock the schematic or symbol before executing the command.

- pDispatch

  Client object which gets invoked when the command is executed. When the command is issued: If the object has a method called OnExecuteCommand, it will be passed the Application object's dispatch pointer, the value of the CommandName parameter, and the parameters passed to the command as a string like this:

  OnExecuteCommand(pViewDrawApp, Command, RestOfLine)

  Otherwise, the Object's method named the same as the CommandName parameter will be called like this:

  CommandName(RestOfLine)

Return Values

As Boolean. True - The command was successfully registered. False - The command could not be registered.

Description

This method enables you to add your own commands to the command line. Command names must be unique and descriptions should be clear. If the command modifies the active block data bModifiesData should be True. You must also pass an Object (dispatch pointer) to your automation entry point that performs the command.

Examples

C++ Command Handler Prototype and Scripting.

```
C++

   void CMyApp::OnExecuteCommand(LPDISPATCH pViewDrawApp, LPCTSTR Command,
 LPCTSTR RestOfLine)
   {

   }

VBSCRIPT

   RegisterOLECommand "foo", "Foo displays a message box", False,
 ScriptEngine

   Sub Foo(RestOfLine)
       MsgBox "Foo Command Executed, RestOfLine=" & RestOfLine
   End Sub
```

UnregisterOLECommand Method (CommandsManager Object)

Scope: Schematic editor

Object: CommandsManager Object

Prerequisites: None

Revokes a previously registered command.

Usage

CommandsManager.**UnregisterOLECommand**(ByVal *CommandName* As String, ByVal *ClientDispatch* As Object) As Boolean

Arguments

- CommandName

  This is the unique name for the command.
- ClientDispatch

  Client object which gets invoked when the command is executed.

Return Values

As Boolean. True - The command was successfully unregistered. False - The command could not be unregistered.

Component Object

This object represents a component placed on a schematic.

The following table lists methods and properties of the Component object with links to the respective reference pages:

Table 13. Component Object Methods and Properties

Method or Property	Description
AddAttribute Method (Component Object)	Adds an attribute to the component at the coordinates passed.
AddBatchAttributes Method (Component Object)	Adds multiple attributes to the component as an encoded string.
AddBatchOats Method (Component Object)	Adds multiple instance values (formerly known as OATs) to the component as an encoded string.
AddLabel Method (Component Object)	Adds a label to the component at the coordinates passed.
AddOat Method (Component Object)	Sets the instance value (formerly referred to as an OAT) for a component attribute.
FindAttribute Method (Component Object)	Searches all component attributes for one matching name.
GetBatchAttributes Method (Component Object)	Returns all attributes on the component as an encoded string.
GetBatchOats Method (Component Object)	Returns all instance values (OATs) on the component as an encoded string.
GetBboxPoint Method (Component Object)	Returns the coordinates of a specified corner of a bounding box.
GetConnections Method (Component Object)	Returns a collection of Connection Objects associated with component.
GetForwardPCB Method (Component Object)	Indicates whether the component should be forward annotated to the PCB.
GetLocation Method (Component Object)	Returns the location coordinates (X,Y) for the component.
GetName Method (Component Object)	Returns the hierarchical path to the component.
SetLocation Method (Component Object)	Specifies the location for the component.
Application Property (Component Object)	Returns the Application Object.
Attributes Property (Component Object)	Returns a collection of Attribute Objects representing the attributes attached to this component.
Id Property (Component Object)	Returns a unique identifying integer for this component.
Label Property (Component Object)	Returns a label object.
Orientation Property (Component Object)	Returns or sets the component orientation (rotation).
Parent Property (Component Object)	Returns the parent Block Object of the component.

Table 13. Component Object Methods and Properties (continued)

Method or Property	Description
Refdes Property (Component Object)	Returns or sets the PCB reference designator for the component.
Scale Property (Component Object)	Returns or sets the scale factor of the component.
Selected Property (Component Object)	Sets the selection status of the component.
SymbolBlock Property (Component Object)	Returns the symbol which represents the component instance.
Type Property (Component Object)	Returns the type for the component.
UID Property (Component Object)	Returns the unique identifying string (UID) of this component.

AddAttribute Method (Component Object)

Scope: Schematic editor

Object: Component Object

Prerequisites: None

Adds an attribute to the component at the coordinates passed.

Usage

Component.**AddAttribute**(ByVal *text* As String, ByVal *X* As Long, ByVal *Y* As Long, ByVal *Visibility* As VdVisibilityFlag) As IVdAttr

Arguments

- text

  Attribute string of the form NAME=VALUE.
- X

  X coordinate of the attribute.
- Y

  Y coordinate of the attribute.
- Visibility

  The visibility of the attribute in the form VdVisibilityFlag Enum.

Return Values

As IVdAttr. The newly created Attribute Object.

AddBatchAttributes Method (Component Object)

Scope: Schematic editor

Object: Component Object

Prerequisites: None

Adds multiple attributes to the component as an encoded string.

Usage

Component.**AddBatchAttributes**(ByVal *AttributeListSting* As String) As Boolean

Arguments

- AttributeListString

  Encoded string containing attributes to be added. The AttributeListString is specially formatted as shown below.

  ```
  <visibility> <duplicate> Name = Value<CR>
  (repeated for each attribute)
  ```

  <*visibility*> should be one of the following integers

  - **0** — Invisible
  - **1** — Name and value visible
  - **2** — Name only visible
  - **3** — Value only visible

  <*duplicate*> should be one of the following integers

  - **0** — add - add new attribute regardless of duplicates.
  - **1** — replace - if attribute with this name exists, replace it

  The <CR> represents a carriage return character (ASCII 13).

Return Values

As Boolean:

- **True** — The attributes were added successfully.
- **False** — The attributes could not be added.

Examples

This example adds two attributes to the component.

```
char* string="0 1 FOO=BAR\r1 1 TEST=BATCH"
pDisp->AddBatchAttributes(string);
```

AddBatchOats Method (Component Object)

Scope: Schematic editor

Object: Component Object

Prerequisites: None

Adds multiple instance values (formerly known as OATs) to the component as an encoded string.

Usage

Component.**AddBatchOats**(ByVal *AttributeListSting* As String) As Boolean

Arguments

- AttributeListSting

  Encoded string containing oats to be added. The AttributeListString is specially formatted as shown below.

  ```
  <visibility> <duplicate> Name=Value<CR>
  (repeated for each attribute)
  ```

  <*visibility*> should be one of the following integers:

  - **0** — Invisible
  - **1** — Name and Value Visible
  - **2** — Name Only Visible
  - **3** — Value Only Visible

  <*duplicate*> should be one of the following integers:

  - **0** — add - add new attribute regardless of duplicates.
  - **1** — replace - if attribute with this name exists, replace it.

  The <CR> represents a carriage return character (ASCII 13).

Return Values

As Boolean:

- **True** — The instance values (OATs) were successfully added.
- **False** — The instance values (OATs) could not be added.

Examples

This example adds instance values to a component.

```
char* string="0 0 FOO=BAR\r1 1 TEST=BATCH"
pDisp->AddBatchOats(string);
```

AddLabel Method (Component Object)

Scope: Schematic editor

Object: Component Object

Prerequisites: None

Adds a label to the component at the coordinates passed.

Usage

Component.**AddLabel**(ByVal *text* As String, ByVal *X* As Long, ByVal *Y* As Long) As IVdLabel

Arguments

- text

  This argument specifies the text contents of the label.
- X

  X coordinate of the label.
- Y

  Y coordinate of the label.

Return Values

As IVdLabel. The newly created Label Object.

AddOat Method (Component Object)

Scope: Schematic editor

Object: Component Object

Prerequisites: None

Sets the instance value (formerly referred to as an OAT) for a component attribute.

Note:
If there is no attribute that matches the String argument, a new attribute is created. You can determine which attribute instance value is being set with the "InstanceValue Property (Attribute Object)" on page 177, which serves as a better alternative for this operation.

Usage

Component.**AddOat**(ByVal *text* As String) As IVdAttr

Arguments

- text

  The attribute instance value string, taking the form: NAME=VALUE

Return Values

As Attribute. The newly created Attribute Object.

FindAttribute Method (Component Object)

Scope: Schematic editor

Object: Component Object

Prerequisites: None

Searches all component attributes for one matching name.

Usage

Component.**FindAttribute**(ByVal *AttributeString* As String) As IVdAttr

Arguments

- AttributeString

  This is the attribute name for which the search is executed.

Return Values

As IVdAttr. The Attribute Object.

Description

This method only finds attributes attached to the component. It does not find attributes that are only on the underlying symbol.

Examples

This example locates an attribute named MODEL.

```
Set Attr = Comp.FindAttribute("MODEL")
If Attr Is Nothing Then
    MsgBox "No MODEL atribute found"
Else
    MsgBox "MODEL=" & Attr.Value
End If
```

GetBatchAttributes Method (Component Object)

Scope: Schematic editor

Object: Component Object

Prerequisites: None

Returns all attributes on the component as an encoded string.

Usage

Component.**GetBatchAttributes**() As String

Arguments

None

Return Values

As String. An encoded string containing the attributes and their values.

Description

The string returned is specially formatted as shown below.

```
Visibility Name=Value<CR>
(repeated for each attribute)
```

Where Visibility is an integer represented by VdVisibilityFlag Enum and the <CR> represents a carriage return character (ASCII 13).

GetBatchOats Method (Component Object)

Scope: Schematic editor

Object: Component Object

Prerequisites: None

Returns all instance values (OATs) on the component as an encoded string.

Usage

Component.**GetBatchOats**() As String

Arguments

None

Return Values

As String. The encoded string containing the OATs and their values. The string returned is specially formatted as follows:

```
Visibility Name=Value<CR>
(repeated for each attribute)
```

Where Visibility is an integer represented by VdVisibilityFlag Enum and the <CR> represents a carriage return character (ASCII 13).

Examples

```
For Each MyComp In ActiveView.Query(VDM_COMP, VD_ALL) Msg("Instance Value
 attribute: " & myComp.GetBatchOats()) Next
```

GetBboxPoint Method (Component Object)

Scope: Schematic editor

Object: Component Object

Prerequisites: None

Returns the coordinates of a specified corner of a bounding box.

Usage

Component.**GetBboxPoint**(ByVal *Location* As VdCorner) As IVdPoint

Arguments

Note:
All coordinates are measured in 100ths of an inch.

- *Location*

  Specifies which location coordinates to return, as described in "VdCorner Enum" on page 712.

Return Values

As IVdPoint. The Point Object representing the coordinates.

Examples

This example displays both coordinates.

```
MsgBox "LowerLeft X=" & Component.GetBboxPoint(VDLOWERLEFT).X & " Y=" &
 Component.GetBboxPoint(VDUPPERRIGHT).Y
```

GetConnections Method (Component Object)

Scope: Schematic editor

Object: Component Object

Prerequisites: None

Returns a collection of Connection Objects associated with component.

Usage

Component.**GetConnections**() As IVdObjs

Arguments

None

Return Values

As IVdObjs. The collection of Connection Objects.

Description

A component may be connected to another by attaching nets (Net Object) to the component pins. A Connection Object is associated with each component pin (ComponentPin Object).

Examples

This example shows all nets attached to the component.

```
For Each Conn In Comp.GetConnections
    Set Net = Conn.Net
    Set CmpPin = Conn.CompPin
    If Not Net Is Nothing Then
       MsgBox "Net Attached to " & CmpPin.Pin.Label.TextString
    End If
Next
```

GetForwardPCB Method (Component Object)

Scope: Schematic editor

Object: Component Object

Prerequisites: None

Indicates whether the component should be forward annotated to the PCB.

Usage

Component.**GetForwardPCB**() As VdCompInstanceForwardPCB

Arguments

None

Return Values

As VdCompInstanceForwardPCB. An enumerator value that represents whether the component should be forward annotated to the PCB. See VdCompInstanceForwardPCB Enum for more information.

GetLocation Method (Component Object)

Scope: Schematic editor

Object: Component Object

Prerequisites: None

Returns the location coordinates (X,Y) for the component.

Usage

Component.**GetLocation**() As IVdPoint

Arguments

None

Return Values

Note:
All coordinates are measured in 100ths of an inch.

As IVdPoint. The Point Object representing coordinates.

GetName Method (Component Object)

Scope: Schematic editor

Object: Component Object

Prerequisites: None

Returns the hierarchical path to the component.

Usage

Component.**GetName**(ByVal *Flag* As VdNameType) As String

Arguments

- Flag

  This argument determines whether the returned value is a fully qualified path or a short name. This is of the form VdNameType Enum.

Return Values

As String. A string containing the fully qualified path or short name.

If you have "pushed" down into the schematic which contains this component and the VdNameType Enum is FULL_PATH_NAME, then the fully qualified path (e.g. $1I1/$1I2) returns. Otherwise, the short name returns.

SetLocation Method (Component Object)

Scope: Schematic editor

Object: Component Object

Prerequisites: None

Specifies the location for the component.

Usage

Component.**SetLocation**(ByVal *X* As Long, ByVal *Y* As Long)

Arguments

Note:
All coordinates are measured in 100ths of an inch.

- X

  X coordinate for the component.
- Y

  Y coordinate for the component.

Application Property (Component Object)

Scope: Schematic editor

Object: Component Object

Access: Read-Only

Prerequisites: None

Returns the Application Object.

Usage

Component.**Application**

Arguments

None

Return Values

IVdApp. The Application Object.

Description

See Application Object for more information.

Attributes Property (Component Object)

Scope: Schematic editor

Object: Component Object

Access: Read-Only

Prerequisites: None

Returns a collection of Attribute Objects representing the attributes attached to this component.

Usage

Component.**Attributes**

Arguments

None

Return Values

IVdObjs. A collection of the Attribute Objects associated with the component.

Description

Attributes are included in the collection whether visible or not. This collection includes only component attributes -- it doesn't include attributes that are only on the underlying symbol.

See Attribute Object for more information.

Id Property (Component Object)

Scope: Schematic editor

Object: Component Object

Access: Read-Only

Prerequisites: None

Returns a unique identifying integer for this component.

Usage

Component.**Id**

Arguments

None

Return Values

Long. A value representing the unique identifier for this component.

Label Property (Component Object)

Scope: Schematic editor

Object: Component Object

Access: Read-Only

Prerequisites: None

Returns a label object.

Usage

Component.**Label**

Arguments

None

Return Values

IVdLabel. The Label Object for this component. If no label is present on the component, a NULL string returns.

Examples

This example tests for the existence of a label.

```
If Component.Label Is Nothing Then
    MsgBox "No Label was Assigned"
Else
    Msgbox "Label is Present and Is= "& Label.TextString
End If
```

Orientation Property (Component Object)

Scope: Schematic editor

Object: Component Object

Access: Read/Write

Prerequisites: None

Returns or sets the component orientation (rotation).

Usage

Component.**Orientation** = VdOrientation

Arguments

None

Return Values

VdOrientation. The return/set type for this property. This is of the form VdOrientation Enum.

Parent Property (Component Object)

Scope: Schematic editor

Object: Component Object

Access: Read-Only

Prerequisites: None

Returns the parent Block Object of the component.

Usage

Component.**Parent**

Arguments

None

Return Values

IVdBlock. The parent Block Object of the component.

Refdes Property (Component Object)

Scope: Schematic editor

Object: Component Object

Access: Read/Write

Prerequisites: None

Returns or sets the PCB reference designator for the component.

Usage

Component.**Refdes** = String

Arguments

None

Return Values

String. A string containing the PCB reference designator for the component.

Scale Property (Component Object)

Scope: Schematic editor

Object: Component Object

Access: Read/Write

Prerequisites: None

Returns or sets the scale factor of the component.

Usage

Component.**Scale** = Double

Arguments

None

Return Values

Double. A value representing the scale factor of the component.

Selected Property (Component Object)

Scope: Schematic editor

Object: Component Object

Access: Write-Only

Prerequisites: None

Sets the selection status of the component.

Usage

Component.**Selected** = True | False

Arguments

None

Return Values

True | False. The set type for this property. True - the component is selected. False - the component is deselected.

SymbolBlock Property (Component Object)

Scope: Schematic editor

Object: Component Object

Access: Read-Only

Prerequisites: None

Returns the symbol which represents the component instance.

Usage

Component.**SymbolBlock**

Arguments

None

Return Values

IVdBlock. A Block Object representing the symbol for the component instance.

Type Property (Component Object)

Scope: Schematic editor

Object: Component Object

Access: Read-Only

Prerequisites: None

Returns the type for the component.

Usage

Component.**Type**

Arguments

None

Return Values

VdObjectType. The return type for this property. This is of the form VdObjectType Enum.

UID Property (Component Object)

Scope: Schematic editor

Object: Component Object

Access: Read-Only

Prerequisites: None

Returns the unique identifying string (UID) of this component.

Usage

Component.**UID**

Arguments

None

Return Values

String. A string containing the UID for the component in the form $<*sheet number*>I<*ID number*> (for example: ($1I34)).

ComponentPin Object

This object represents a pin associated with a component on a schematic.

The following table lists methods and properties of the ComponentPin object with links to the respective reference pages:

Table 14. ComponentPin Object Methods and Properties

Method or Property	Description
AddAttribute Method (ComponentPin Object)	Adds an attribute to the component pin at the coordinates passed.
AddOAT Method (ComponentPin Object)	Sets the instance value (OAT) of a component pin attribute.
FindAttribute Method (ComponentPin Object)	Searches all component pin attributes for one matching attribute name.
GetLocation Method (ComponentPin Object)	Returns a point object containing component pin coordinates.
Application Property (ComponentPin Object)	Returns the Application Object.
Attributes Property (ComponentPin Object)	Returns a collection of all Attribute Objects on this component pin whether they are visible or not.
Component Property (ComponentPin Object)	Returns the Component Object associated with the component pin.
Connection Property (ComponentPin Object)	Returns the Connection Object associated with the component pin.
Number Property (ComponentPin Object)	Returns or sets the component PCB pin number identifier.
Parent Property (ComponentPin Object)	Returns the parent Component Object for the component pin object.
Pin Property (ComponentPin Object)	Returns the Pin Object associated with the component pin.
Selected Property (ComponentPin Object)	Sets the selection status for the component pin.
Side Property (ComponentPin Object)	Returns the side of the component on which the component pin exists.
Type Property (ComponentPin Object)	Returns the component pin type.

AddAttribute Method (ComponentPin Object)

Scope: Schematic editor

Object: ComponentPin Object

Prerequisites: None

Adds an attribute to the component pin at the coordinates passed.

Usage

ComponentPin.**AddAttribute**(ByVal *text* As String, ByVal *X* As Long, ByVal *Y* As Long, ByVal *Visibility* As VdVisibilityFlag) As IVdAttr

Arguments

- *text*

  Attribute string of the form NAME=VALUE.
- *X*

  X coordinate of attribute.
- *Y*

  Y coordinate of attribute.
- *Visibility*

  This argument specifies the visibility of the attribute in the form VdVisibilityFlag Enum.

Return Values

As IVdAttr. The newly created Attribute Object.

AddOAT Method (ComponentPin Object)

Scope: Schematic editor

Object: ComponentPin Object

Prerequisites: None

Sets the instance value (OAT) of a component pin attribute.

Usage

ComponentPin.**AddOat**(ByVal *String* As String) As IVdAttr

Arguments

- String

  Attribute string of the form NAME=VALUE.

Return Values

As IVdAttr. The Attribute Object that has the new OAT value.

Description

Sets the instance value (OAT) of a component pin attribute. If no attribute with the given name exists, creates a new attribute. You can control the context (fully qualified path) by using the SelectPath Method (Application Object).

FindAttribute Method (ComponentPin Object)

Scope: Schematic editor

Object: ComponentPin Object

Prerequisites: None

Searches all component pin attributes for one matching attribute name.

Usage

ComponentPin.**FindAttribute**(ByVal *AttributeName* As String) As IVdAttr

Arguments

- AttributeName

  The attribute name that is targeted by the search.

Return Values

As IVdAttr. The attribute object, if found; otherwise, the NULL string is returned.

Examples

This example finds the PINTYPE attribute.

```
Set Attr = CompPin.FindAttribute("PINTYPE")
If Attr Is Nothing Then
    MsgBox "No PINTYPE atribute found"
Else
    MsgBox "PINTYPE=" & Attr.Value
End If
```

GetLocation Method (ComponentPin Object)

Scope: Schematic editor

Object: ComponentPin Object

Prerequisites: None

Returns a point object containing component pin coordinates.

Note:
All coordinates are measured in 100ths of an inch.

Usage

ComponentPin.**GetLocation**() As IVdPoint

Arguments

None

Return Values

As IVdPoint. The Point Object representing coordinates of the component pin, which are always located at the bounding box of the component.

Application Property (ComponentPin Object)

Scope: Schematic editor

Object: ComponentPin Object

Access: Read-Only

Prerequisites: None

Returns the Application Object.

Usage

ComponentPin.**Application**

Arguments

None

Return Values

IVdApp. The Application Object for this component pin.

Description

See Application Object for more information.

Attributes Property (ComponentPin Object)

Scope: Schematic editor

Object: ComponentPin Object

Access: Read-Only

Prerequisites: None

Returns a collection of all Attribute Objects on this component pin whether they are visible or not.

Usage

ComponentPin.**Attributes**

Arguments

None

Return Values

IVdObjs. A collection of Attribute Objects on this component pin.

Description

See Attribute Object for more information.

Component Property (ComponentPin Object)

Scope: Schematic editor

Object: ComponentPin Object

Access: Read-Only

Prerequisites: None

Returns the Component Object associated with the component pin.

Usage

ComponentPin.**Component**

Arguments

None

Return Values

IVdComp. The Component Object for this component pin.

Description

See Component Object for more information.

Connection Property (ComponentPin Object)

Scope: Schematic editor

Object: ComponentPin Object

Access: Read-Only

Prerequisites: None

Returns the Connection Object associated with the component pin.

Usage

ComponentPin.**Connection**

Arguments

None

Return Values

IVdConnection. The Connection Object associated with the component pin.

Description

See Connection Object for more information.

Number Property (ComponentPin Object)

Scope: Schematic editor

Object: ComponentPin Object

Access: Read/Write

Prerequisites: None

Returns or sets the component PCB pin number identifier.

Note:
This method only returns a value if the pin number is defined at the instance or block level. If only the symbol level value is present, the pin number identifier is not returned.

Usage

ComponentPin.**Number** = String

Arguments

None

Return Values

String. A string containing the PCB pin number identifier for the component pin. The PCB pin number identifier is usually the value of a #= attribute

Parent Property (ComponentPin Object)

Scope: Schematic editor

Object: ComponentPin Object

Access: Read-Only

Prerequisites: None

Returns the parent Component Object for the component pin object.

Usage

ComponentPin.**Parent**

Arguments

None

Return Values

IVdComp. The parent Component Object for the component pin.

Pin Property (ComponentPin Object)

Scope: Schematic editor

Object: ComponentPin Object

Access: Read-Only

Prerequisites: None

Returns the Pin Object associated with the component pin.

Usage

ComponentPin.**Pin**

Arguments

None

Return Values

IVdPin. The Pin Object for the component pin.

Selected Property (ComponentPin Object)

Scope: Schematic editor

Object: ComponentPin Object

Access: Read-Only

Prerequisites: None

Sets the selection status for the component pin.

Usage

ComponentPin.**Selected** = True | False

Arguments

None

Return Values

True | False. True - Selects the component pin. False - Deselects the component pin.

Side Property (ComponentPin Object)

Scope: Schematic editor

Object: ComponentPin Object

Access: Read-Only

Prerequisites: None

Returns the side of the component on which the component pin exists.

Usage

ComponentPin.**Side**

Arguments

None

Return Values

VdSide. The return type for this property. This is of the form VdSide Enum.

Type Property (ComponentPin Object)

Scope: Schematic editor

Object: ComponentPin Object

Access: Read-Only

Prerequisites: None

Returns the component pin type.

Usage

ComponentPin.**Type**

Arguments

None

Return Values

VdObjectType. The return type for this property. This is of the form VdObjectType Enum.

Connection Object

A connection object represents a component-to-net, net-to-bus, or bus-to-bus connection on a schematic.

The following table lists properties of the Connection object with links to the respective reference pages:

Table 15. Connection Object Properties

Property	Description
CompPin Property (Connection Object)	Returns the ComponentPinObject for this connection.
Net Property (Connection Object)	Returns the Net Object for the connection.
Ripper Property (Connection Object)	Returns the Ripper Object for the connection.
Segment Property (Connection Object)	Returns the Segment Object associated with this connection.

CompPin Property (Connection Object)

Scope: Schematic editor

Object: Connection Object

Access: Read-Only

Prerequisites: None

Returns the ComponentPinObject for this connection.

Usage

Connection.**CompPin**

Arguments

None

Return Values

IVdCmpPin. The ComponentPin Object.

Description

See ComponentPin Object for more information.

Net Property (Connection Object)

Scope: Schematic editor

Object: Connection Object

Access: Read-Only

Prerequisites: None

Returns the Net Object for the connection.

Usage

Connection.**Net**

Arguments

None

Return Values

IVdNet. The Net Object.

Ripper Property (Connection Object)

Scope: Schematic editor

Object: Connection Object

Access: Read-Only

Prerequisites: None

Returns the Ripper Object for the connection.

Usage

Connection.Ripper

Arguments

None

Return Values

Ripper. The Ripper Object.

Segment Property (Connection Object)

Scope: Schematic editor

Object: Connection Object

Access: Read-Only

Prerequisites: None

Returns the Segment Object associated with this connection.

Usage

Connection.**Segment**

Arguments

None

Return Values

IVdSegment. The Segment Object.

DesignerDataManagement Object

You can access EDM managed designs using the DesignerDataManagement object. The following table lists methods and properties of the DesignerDataManagement object with links to the respective reference pages.

Table 16. DesignerDataManagement Object Methods and Properties

Method or Property	Description
AdditionalProperties Method (DesignerDataManagement Object)	Find additional properties available for entities of given types.
Connect Method (DesignerDataManagement Object)	Connects to a data management server.
entities Method (DesignerDataManagement Object)	Finds all entities.
FindEntityByPath Method (DesignerDataManagement Object)	Finds an entity by its path.
FindEntityByPathAndVersion Method (DesignerDataManagement Object)	Finds an entity by its path and version.
IsConnected Method (DesignerDataManagement Object)	Checks the data management connection status.
LastErrorMessage Method (DesignerDataManagement Object)	Returns the most recent error message.
OpenComponentView Method (DesignerDataManagement Object)	Opens the component view.
OpenManagedBlockView Method (DesignerDataManagement Object)	Opens the managed block view.
RecentEntities Method (DesignerDataManagement Object)	Returns recently used entities.
ValidEntityTypes Method (DesignerDataManagement Object)	Returns the valid entity types.
Option Property (DesignerDataManagement Object)	Returns the data management related option value.
OptionIsReadOnly Property (DesignerDataManagement Object)	Checks whether a given data management related option is read-only.

AdditionalProperties Method (DesignerDataManagement Object)

Scope: Schematic editor; Embedded Client (EDM)

Prerequisites: None

Object: DesignerDataManagement Object

Find additional properties available for entities of given types.

Usage

DesignerDataManagement,**AdditionalProperties**(ByVal *Type* As VdDMEntityType) As IMGCDesignerDataManagementAdditionalProperties

Arguments

- Type

  Enumerator that specifies the data management entity type (VdDMEntityType Enum).

Return Values

IMGCDesignerDataManagementAdditionalProperties. Returns a collection of additional properties (“DesignerDataManagementAdditionalProperties” on page 644) for the given type.

Connect Method (DesignerDataManagement Object)

Scope: Schematic editor; Embedded Client (EDM)

Object: DesignerDataManagement Object

Prerequisites: None

Connects to a data management server.

Usage

DesignerDataManagement.**Connect**(ByVal server_address As String, ByVal server_port As String) As Nothing

Arguments

- server_address

  A string that contains the address of the data management server.
- server_port

  A string that contains the port number of the data management server.

Return Values

Nothing

entities Method (DesignerDataManagement Object)

Scope: Schematic editor; Embedded Client (EDM)

Object: DesignerDataManagement Object

Prerequisites: None

Finds all entities.

Usage

DesignerDataManagement.**entities**(ByVal Type As VdDMEntityType) As IMGCDesignerDataManagementEntities

Arguments

- Type

  Enumerator that specifies the data management entity type to return. See VdDMEntityType Enum for more information.

Return Values

IMGCDesignerDataManagementEntities. Returns a collection of all entities (“DesignerDataManagementEntities” on page 648).

FindEntityByPath Method (DesignerDataManagement Object)

Scope: Schematic editor; Embedded Client (EDM)

Object: DesignerDataManagement Object

Prerequisites: None

Finds an entity by its path.

Usage

DesignerDataManagement.**FindEntityByPath**(ByVal Path As String) As IMGCDesignerDataManagementEntity

Arguments

- Path

  A String that contains the data management entity's path.

Return Values

IMGCDesignerDataManagementEntity. Returns an entity ("DesignerDataManagementEntity" on page 354).

FindEntityByPathAndVersion Method (DesignerDataManagement Object)

Scope: Schematic editor; Embedded Client (EDM)

Object: DesignerDataManagement Object

Prerequisites: None

Finds an entity by its path and version.

Usage

```
DesignerDataManagement.FindEntityByPathAndVersion(
        ByVal Path As String,
        ByVal Version As String) As IMGCDesignerDataManagementEntity
```

Arguments

- Path

  A String that contains the data management entity's path.
- Version

  A String that contains the data management entity's version.

Return Values

IMGCDesignerDataManagementEntity. Returns an entity ("DesignerDataManagementEntity" on page 354).

IsConnected Method (DesignerDataManagement Object)

Scope: Schematic editor; Embedded Client (EDM)

Object: DesignerDataManagement Object

Prerequisites: None

Checks the data management connection status.

Usage

DesignerDataManagement.**IsConnected**() As Boolean

Arguments

None

Return Values

Boolean. If True, the data management object is connected. If False, it is not connected.

LastErrorMessage Method (DesignerDataManagement Object)

Scope: Schematic editor; Embedded Client (EDM)

Object: DesignerDataManagement Object

Prerequisites: None

Returns the most recent error message.

Usage

DesignerDataManagement.**LastErrorMessage**() As String

Arguments

None

Return Values

String. Returns a string containing the last error message.

OpenComponentView Method (DesignerDataManagement Object)

Scope: Schematic editor; Embedded Client (EDM)

Object: DesignerDataManagement Object

Prerequisites: None

Opens the component view.

Usage

DesignerDataManagement.**OpenComponentView**(ByVal partNumber As String) As Nothing

Arguments

- partNumber

  A String that contains the part number for the component.

Return Values

Nothing

OpenManagedBlockView Method (DesignerDataManagement Object)

Scope: Schematic editor; Embedded Client (EDM)

Object: DesignerDataManagement Object

Prerequisites: None

Opens the managed block view.

Usage

DesignerDataManagement.**OpenManagedBlockView**(
ByVal Name As String,
ByVal Version As String) As Nothing

Arguments

- Name

  A String that contains the name of the managed block.
- Version

  A String that contains the version of the managed block.

Return Values

Nothing

RecentEntities Method (DesignerDataManagement Object)

Scope: Schematic editor; Embedded Client (EDM)

Object: DesignerDataManagement Object

Prerequisites: None

Returns recently used entities.

Usage

DesignerDataManagement.**RecentEntities**() As IMGCDesignerDataManagementEntities

Arguments

None

Return Values

IMGCDesignerData ManagementEntities. Returns a collection of recently used entities (“DesignerDataManagementEntities” on page 648).

ValidEntityTypes Method (DesignerDataManagement Object)

Scope: Schematic editor; Embedded Client (EDM)

Object: DesignerDataManagement Object

Prerequisites: None

Returns the valid entity types.

Usage

DesignerDataManagement.**ValidEntityTypes**() As VdDMEntityType

Arguments

None

Return Values

VdDMEntityType. Returns an enumerator that specifies the data management entity type. See VdDMEntityType Enum for more information.

Option Property (DesignerDataManagement Object)

Scope: Schematic editor; Embedded Client (EDM)

Object: DesignerDataManagement Object

Access: Read/Write

Prerequisites: None

Returns the data management related option value.

Usage

DesignerDataManagement.**Option**(ByVal Name As String) = String

Arguments

- Name

  String that represents the related option.

Return Values

String. Returns a string containing the data management related option value.

OptionIsReadOnly Property (DesignerDataManagement Object)

Scope: Schematic editor; Embedded Client (EDM)

Object: DesignerDataManagement Object

Access: Read/Write

Prerequisites: None

Checks whether a given data management related option is read-only.

Usage

DesignerDataManagement.**OptionIsReadOnly**(ByVal Name As String) = True | False

Arguments

- Name

  String that represents the related option.

Return Values

True | False. If True, the related option is read-only. If False, the related option is not read-only.

DesignerDataManagementAdditionalProperty Object

You can access EDM managed designs using the DesignerDataManagementAdditionalProperty object. The following table lists methods and properties of the DesignerDataManagementAdditionalProperty object with links to the respective reference pages.

Table 17. DesignerDataManagementAdditionalProperty Object Methods and Properties

Method or Property	Description
id Method (DesignerDataManagementAdditionalProperty Object)	Returns the id of the additional property.
Label Method (DesignerDataManagementAdditionalProperty Object)	Returns the label of the additional property.

id Method (DesignerDataManagementAdditionalProperty Object)

Scope: Schematic editor; Embedded Client (EDM)

Object: DesignerDataManagementAdditionalProperty Object

Prerequisites: None

Returns the id of the additional property.

Usage

DesignerDataManagementAdditionalProperty.**id**() As String

Arguments

None

Return Values

String. A string that contains the id of the Designer data management additional property.

Label Method (DesignerDataManagementAdditionalProperty Object)

Scope: Schematic editor; Embedded Client (EDM)

Object: DesignerDataManagementAdditionalProperty Object

Prerequisites: None

Returns the label of the additional property.

Usage

DesignerDataManagementAdditionalProperty.**Label**() As String

Arguments

None

Return Values

String. A string that contains the label of the data management additional property.

DesignerDataManagementEntity Object

You can access EDM managed designs using the DesignerDataManagementEntity object. The following table lists methods and properties of the DesignerDataManagementEntity object with links to the respective reference pages.

Table 18. DesignerDataManagementEntity Object Methods and Properties

Method or Property	Description
AccessType Method (DesignerDataManagementEntity Object)	Returns the access type of the data management entity.
AdditionalPropertyValue Method (DesignerDataManagementEntity Object)	Returns the value of an additional property available on the entity.
CancelEditing Method (DesignerDataManagementEntity Object)	Cancels editing of the entity.
CheckedInDate Method (DesignerDataManagementEntity Object)	Returns the checked-in date of the data management entity.
CheckIn Method (DesignerDataManagementEntity Object)	Checks in a checked-out entity. The entity cannot be actively edited.
DumpProperties Method (DesignerDataManagementEntity Object)	Dumps EDM Design object metadata to an output file in JSON format.
EditShared Method (DesignerDataManagementEntity Object)	Opens the entity for edit shared.
EditExclusive Method (DesignerDataManagementEntity Object)	Opens entity for edit exclusive.
Export Method (DesignerDataManagementEntity Object)	Exports the entity to the passed destination.
LaunchWebView Method (DesignerDataManagementEntity Object)	Launches a web view for the entity.
Name Method (DesignerDataManagementEntity Object)	Returns the name for the data management entity object.
OpenReference Method (DesignerDataManagementEntity Object)	Opens the reference of the entity.
Path Method (DesignerDataManagementEntity Object)	Returns the path of the data management entity.
ProjectName Method (DesignerDataManagementEntity Object)	Returns the project name of the data management entity.

Table 18. DesignerDataManagementEntity Object Methods and Properties (continued)

Method or Property	Description
PropertyValueById Method (DesignerDataManagementEntity Object)	Returns the value of a property that is available on the entity.
RelatedEntities Method (DesignerDataManagementEntity Object)	Finds all related entities.
RunConfigurationRules Method (DesignerDataManagementEntity Object)	Runs the entity configuration rules.
Status Method (DesignerDataManagementEntity Object)	Returns the status of the data management entity.
Type Method (DesignerDataManagementEntity Object)	Returns the type of the entity.
Unlock Method (DesignerDataManagementEntity Object)	Unlocks the entity.
Users Method (DesignerDataManagementEntity Object)	Returns the users of the data management entity.
Version Method (DesignerDataManagementEntity Object)	Returns the version of the data management entity.
View Method (DesignerDataManagementEntity Object)	Opens the entity for view.
ViewForOutputsGeneration Method (DesignerDataManagementEntity Object)	Opens the entity for view and outputs generation.

AccessType Method (DesignerDataManagementEntity Object)

Scope: Schematic editor; Embedded Client (EDM)

Object:DesignerDataManagementEntity Object

Prerequisites: None

Returns the access type of the data management entity.

Usage

DesignerDataManagementEntity.**AccessType**() As VdDMEntityAccessType

Arguments

None

Return Values

As VdDMEntityAccessType. An enumerator type that represents the entity's access type. See VdDMEntityAccessType Enum for more information.

AdditionalPropertyValue Method (DesignerDataManagementEntity Object)

Scope: Schematic editor; Embedded Client (EDM)

Object:DesignerDataManagementEntity Object

Prerequisites: None

Returns the value of an additional property available on the entity.

Usage

DesignerDataManagementEntity.**AdditionalPropertyValue**(ByVal property As IMGCDesignerDataManagementAdditionalProperty) As String

Arguments

- property

  IMGCDesignerDataManagementAdditionalProperty. A data management additional property available on the entity.

Return Values

String. A string that contains the value of the given additional property.

CancelEditing Method (DesignerDataManagementEntity Object)

Scope: Schematic editor; Embedded Client (EDM)

Object: DesignerDataManagementEntity Object

Prerequisites: None

Cancels editing of the entity.

Usage

DesignerDataManagementEntity.**CancelEditing**() As Nothing

Arguments

None

Return Values

Nothing

CheckedInDate Method (DesignerDataManagementEntity Object)

Scope: Schematic editor; Embedded Client (EDM)

Object:DesignerDataManagementEntity Object

Prerequisites: None

Returns the checked-in date of the data management entity.

Usage

DesignerDataManagementEntity.**CheckedInDate**() As String

Arguments

None

Return Values

As String. A string that contains the entity's checked-in date.

CheckIn Method (DesignerDataManagementEntity Object)

Scope: Schematic editor; Embedded Client (EDM)

Object:DesignerDataManagementEntity Object

Prerequisites: None

Checks in a checked-out entity. The entity cannot be actively edited.

Usage

DesignerDataManagementEntity.**CheckIn**(ByVal comment As String) As Nothing

Arguments

- comment

  As String. A string containing a comment.

Return Values

Nothing

DumpProperties Method (DesignerDataManagementEntity Object)

Scope: Schematic editor; Embedded Client (EDM)

Object:DesignerDataManagementEntity Object

Prerequisites: None

Dumps EDM Design object metadata to an output file in JSON format.

Usage

DesignerDataManagementEntity.**DumpProperties**(
ByVal outputDirectory As String,
[ByVal filter As VdDMDumpPropertiesFilter = VDDMDUMPFILTER_ALL],
[ByVal includeParents As Boolean = False]) As String

Arguments

- outputDirectory

  String. A string containing the path to the generated JSON file.

- filter

  (Optional) VdDMDumpPropertiesFilter. An enum that represents the type of EDM metadata being dumped to the output JSON file. The default is 0 (all metadata). See VdDMDumpPropertiesFilter Enum for more information.

- includeParents

  (Optional) Boolean. Specifies whether to include parents metadata. The default is False.

Return Values

String. Returns the string containing the output JSON file path.

EditShared Method (DesignerDataManagementEntity Object)

Scope: Schematic editor; Embedded Client (EDM)

Object:DesignerDataManagementEntity Object

Prerequisites: None

Opens the entity for edit shared.

Usage

DesignerDataManagementEntity.**EditShared**() As Boolean

Arguments

None

Return Values

As Boolean. True - the entity was opened for edit shared successfully. False - the entity could not be opened for edit shared.

EditExclusive Method (DesignerDataManagementEntity Object)

Scope: Schematic editor; Embedded Client (EDM)

Object:DesignerDataManagementEntity Object

Prerequisites: None

Opens entity for edit exclusive.

Usage

DesignerDataManagementEntity.**EditExclusive**() As Boolean

Arguments

None

Return Values

As Boolean. True - the entity was opened for edit exclusive. False - the entity could not be opened for edit exclusive.

Export Method (DesignerDataManagementEntity Object)

Scope: Schematic editor; Embedded Client (EDM)

Object:DesignerDataManagementEntity Object

Prerequisites: None

Exports the entity to the passed destination.

Usage

DesignerDataManagementEntity.**Export**(ByVal destination As String) As Nothing

Arguments

- destination

  A string that contains the export destination.

Return Values

Nothing

LaunchWebView Method (DesignerDataManagementEntity Object)

Scope: Schematic editor; Embedded Client (EDM)

Object:DesignerDataManagementEntity Object

Prerequisites: None

Launches a web view for the entity.

Usage

DesignerDataManagementEntity.**LaunchWebView**() As Nothing

Arguments

None

Return Values

Nothing

Name Method (DesignerDataManagementEntity Object)

Scope: Schematic editor; Embedded Client (EDM)

Object:DesignerDataManagementEntity Object

Prerequisites: None

Returns the name for the data management entity object.

Usage

DesignerDataManagementEntity.**Name**() As String

Arguments

None

Return Values

As String. A string containing the name of the entity.

Examples

The following example outputs the entity name:

```
s = s + "Data Management Entity Name: " + DDMEntityObj.Name + vbCrLf
```

OpenReference Method (DesignerDataManagementEntity Object)

Scope: Schematic editor; Embedded Client (EDM)

Object:DesignerDataManagementEntity Object

Prerequisites: None

Opens the reference of the entity.

Usage

DesignerDataManagementEntity.**OpenReference**() As Nothing

Arguments

None

Return Values

Nothing

Path Method (DesignerDataManagementEntity Object)

Scope: Schematic editor; Embedded Client (EDM)

Object:DesignerDataManagementEntity Object

Prerequisites: None

Returns the path of the data management entity.

Note:
Use the "Name Method" on page 366 to retrieve the name of the entity.

Usage

DesignerDataManagementEntity.**Path**() As String

Arguments

None

Return Values

As String. A string that contains the path of the entity.

ProjectName Method (DesignerDataManagementEntity Object)

Scope: Schematic editor; Embedded Client (EDM)

Object:DesignerDataManagementEntity Object

Prerequisites: None

Returns the project name of the data management entity.

Usage

DesignerDataManagementEntity.**ProjectName**() As String

Arguments

None

Return Values

As String. A string that contains the entity's project name.

PropertyValueById Method (DesignerDataManagementEntity Object)

Scope: Schematic editor; Embedded Client (EDM)

Object: DesignerDataManagementEntity Object

Prerequisites: None

Returns the value of a property that is available on the entity.

Usage

DesignerDataManagementEntity.**PropertyValueById**(ByVal *Id* As String) As String

Arguments

- Id

  String that contains the id of a property on the entity.

Return Values

String that contains the value of a property of the entity.

RelatedEntities Method (DesignerDataManagementEntity Object)

Scope: Schematic editor; Embedded Client (EDM)

Object:DesignerDataManagementEntity Object

Prerequisites: None

Finds all related entities.

Usage

DesignerDataManagementEntity.**RelatedEntities**(ByVal types As VdDMEntityType) As IMGCDesignerDataManagementEntities

Arguments

- VdDMEntityAccessType

  An enumerator type that represents the related entities' type. See VdDMEntityAccessType Enum for more information.

Return Values

As IMGCDesignerDataManagementEntities. A collection of IMGCDesignerDataManagementEntity objects.

RunConfigurationRules Method (DesignerDataManagementEntity Object)

Scope: Schematic editor; Embedded Client (EDM)

Object:DesignerDataManagementEntity Object

Prerequisites: None

Runs the entity configuration rules.

Usage

DesignerDataManagementEntity.**RunConfigurationRules**(ByVal detailsFilePath As String) As Nothing

Arguments

- detailsFilePath

  String. A string that contains the filepath to the configuration file.

Return Values

Nothing

Status Method (DesignerDataManagementEntity Object)

Scope: Schematic editor; Embedded Client (EDM)

Object:DesignerDataManagementEntity Object

Prerequisites: None

Returns the status of the data management entity.

Usage

DesignerDataManagementEntity.**Status**() As String

Arguments

None

Return Values

As String. A string that contains the entity's status.

Type Method (DesignerDataManagementEntity Object)

Scope: Schematic editor; Embedded Client (EDM)

Object:DesignerDataManagementEntity Object

Prerequisites: None

Returns the type of the entity.

Usage

DesignerDataManagementEntity.**Type**() As VdDMEntityType

Arguments

None

Return Values

As VdDMEntityType. An enumerator value that represents the entity type. See VdDMEntityType Enum for more information.

Unlock Method (DesignerDataManagementEntity Object)

Scope: Schematic editor; Embedded Client (EDM)

Object: DesignerDataManagementEntity Object

Prerequisites: None

Unlocks the entity.

Usage

DesignerDataManagementEntity.**Unlock**() As Nothing

Arguments

None

Return Values

Nothing

Users Method (DesignerDataManagementEntity Object)

Scope: Schematic editor; Embedded Client (EDM)

Object:DesignerDataManagementEntity Object

Prerequisites: None

Returns the users of the data management entity.

Usage

DesignerDataManagementEntity.**Users**() As String

Arguments

None

Return Values

As String. A string that contains the entity's users.

Version Method (DesignerDataManagementEntity Object)

Scope: Schematic editor; Embedded Client (EDM)

Object:DesignerDataManagementEntity Object

Prerequisites: None

Returns the version of the data management entity.

Usage

DesignerDataManagementEntity.**Version**() As String

Arguments

None

Return Values

As String. A string that contains the version of the entity.

Examples

The following example outputs the version of the data management entity:

```
s = s & "Version: " & DDMEntity.Version & vbCrLf
```

View Method (DesignerDataManagementEntity Object)

Scope: Schematic editor; Embedded Client (EDM)

Object:DesignerDataManagementEntity Object

Prerequisites: None

Opens the entity for view.

Usage

DesignerDataManagementEntity.**View**() As Boolean

Arguments

None

Return Values

As Boolean. True - the entity was opened for view successfully. False - the entity could not be opened for view.

ViewForOutputsGeneration Method (DesignerDataManagementEntity Object)

Scope: Schematic editor; Embedded Client (EDM)

Object:DesignerDataManagementEntity Object

Prerequisites: None

Opens the entity for view and outputs generation.

Usage

DesignerDataManagementEntity.**ViewForOutputsGeneration**() As Boolean

Arguments

None

Return Values

As Boolean. True - the entity was opened for view and outputs generation. False - the entity could not be opened for view and outputs generation.

HDLSourceDocument Object

This object represents an HDL source file document that has been opened in Xpedition Designer.

The following table lists methods and properties of the HDLSourceDocument object with links to the respective reference pages.

Table 19. HDLSourceDocument Object Methods and Properties

Method or Property	Description
BookmarkLine Method (HDLSourceDocument Object)	Adds a bookmark to a specified line in the HDL document.
GotoLine Method (HDLSourceDocument Object)	Moves the cursor to a specified line in the text editor.
Name Property (HDLSourceDocument Object)	Sets or returns the name of the HDLSourceDocument Object.
Path Property (HDLSourceDocument Object)	Sets or returns the path of the HDLSourceDocument Object.

BookmarkLine Method (HDLSourceDocument Object)

Scope: Schematic editor

Object: HDLSourceDocument Object

Prerequisites: None

Adds a bookmark to a specified line in the HDL document.

Usage

HDLSourceDocument.**BookmarkLine**(ByVal *Index* As Long)

Arguments

- Index

  The line number that is assigned the bookmark.

GotoLine Method (HDLSourceDocument Object)

Scope: Schematic editor

Object: HDLSourceDocument Object

Prerequisites: None

Moves the cursor to a specified line in the text editor.

Usage

HDLSourceDocument.**GotoLine**(ByVal *Index* As Long)

Arguments

- Index

  The line number to which the cursor is moved.

Name Property (HDLSourceDocument Object)

Scope: Schematic editor

Object: HDLSourceDocument Object

Access: Read/Write

Prerequisites: None

Sets or returns the name of the HDLSourceDocument Object.

Usage

HDLSourceDocument.**Name** = *String*

Arguments

None

Return Values

String. A string representing the path to the HDLSourceDocument Object.

Path Property (HDLSourceDocument Object)

Scope: Schematic editor

Object: HDLSourceDocument Object

Access: Read/Write

Prerequisites: None

Sets or returns the path of the HDLSourceDocument Object.

Usage

HDLSourceDocument.**Path** = *String*

Arguments

None

Return Values

String. A string representing the path to the HDLSourceDocument Object.

Label Object

The label object represents the label of an object in Xpedition Designer.

The following table lists methods and properties of the Label object with links to the respective reference pages.

Table 20. Label Object Methods and Properties

Method or Property	Description
GetLocation Method (Label Object)	Returns the coordinates of the label as a Point Object.
GetObjectColor Method (Label Object)	Gets the color in which the label is drawn.
IsColorAutomatic Method (Label Object)	Determines if the label has an automatic color set for it.
SetAutomaticColor Method (Label Object)	Sets or unsets the color of a label object as the automatic color for labels.
SetLocation Method (Label Object)	Specifies the coordinates for the label.
SetObjectColor Method (Label Object)	Sets the color in which the label is drawn.
Application Property (Label Object)	Returns the Application Object.
Font Property (Label Object)	Returns or sets the font used for the label.
Orientation Property (Label Object)	Returns or sets the orientation for the label.
Origin Property (Label Object)	Returns or sets the coordinates for the origin of the label.
Parent Property (Label Object)	Returns the parent object for the label (the object on which the label is placed).
ResolvedName Property (Label Object)	Returns the text string of the label and fully resolves any nicknames.
Scope Property (Label Object)	Returns or sets the scope of the label.
Selected Property (Label Object)	Sets the selection status for the label.
Sense Property (Label Object)	Returns or sets the sense of the label that specifies if signal is inverted or not inverted. A label drawn with an overbar indicates an inverted signal.
Size Property (Label Object)	Returns or sets the size (text height) of the label.
TextString Property (Label Object)	Returns or sets the text value of the label.
Type Property (Label Object)	Returns the type value VDTS_LABEL, indicating that the object is, in fact, a label.
Visible Property (Label Object)	Returns or sets the visibility of the label.

GetLocation Method (Label Object)

Scope: Schematic editor

Object: Label Object

Prerequisites: None

Returns the coordinates of the label as a Point Object.

Note:
All coordinates are measured in 100ths of an inch.

Usage

Label.**GetLocation**() As IVdPoint

Arguments

None

Return Values

As IVdPoint. The Point Object.

GetObjectColor Method (Label Object)

Scope: Schematic editor

Object: Label Object

Prerequisites: None

Gets the color in which the label is drawn.

Usage

ActiveColor = *Label*.**GetObjectColor**() As IColor

Arguments

None

Return Values

As Color. See "CColor Object" on page 256.

IsColorAutomatic Method (Label Object)

Scope: Schematic editor

Object: Label Object

Prerequisites: None

Determines if the label has an automatic color set for it.

Usage

Label.**IsColorAutomatic**() As Boolean

Arguments

None

Return Values

As Boolean. True - there is an automatic color set for the label. False - no automatic color is set for the label.

For more information, see “SetAutomaticColor Method (Label Object)” on page 389.

SetAutomaticColor Method (Label Object)

Scope: Schematic editor

Object: Label Object

Prerequisites: None

Sets or unsets the color of a label object as the automatic color for labels.

Usage

Label.**SetAutomaticColor**(byVal *bAutomatic* as Boolean)

Arguments

- bAutomatic

  True - sets the automatic color for the label.

  False - unsets the automatic color for the label.

  If an object has automatic color, as determined by the IsColorAutomatic Method (Label Object), the actual color of the object is the default for this type of object.

SetLocation Method (Label Object)

Scope: Schematic editor

Object: Label Object

Prerequisites: None

Specifies the coordinates for the label.

Note:
All coordinates are measured in 100ths of an inch.

Usage

Label.**SetLocation**(ByVal *X* As Long, ByVal *Y* As Long)

Arguments

- X

  X coordinate
- Y

  Y coordinate

SetObjectColor Method (Label Object)

Scope: Schematic editor

Object: Label Object

Prerequisites: None

Sets the color in which the label is drawn.

Usage

Label.**SetObjectColor**(ByVal *NewColor* as IColor)

Arguments

- Color

  The color assigned to the label. The new color is assigned as a Color object, as described in "CColor Object" on page 256.

Application Property (Label Object)

Scope: Schematic editor

Object: Label Object

Access: Read-Only

Prerequisites: None

Returns the Application Object.

Usage

Label.**Application**

Arguments

None

Return Values

IVdApp. The Application Object.

Description

See Application Object for more information.

Font Property (Label Object)

Scope: Schematic editor

Object: Label Object

Access: Read/Write

Prerequisites: None

Returns or sets the font used for the label.

Usage

Label.**Font** = VdFont

Arguments

None

Return Values

VdFont. The return/set type for this property. This is of the form VdFont Enum.

Orientation Property (Label Object)

Scope: Schematic editor

Object: Label Object

Access: Read/Write

Prerequisites: None

Returns or sets the orientation for the label.

Usage

Label.**Orientation** = VdOrientation

Arguments

None

Return Values

VdOrientation. The return/set type for this property. This is of the form VdOrientation Enum.

Origin Property (Label Object)

Scope: Schematic editor

Object: Label Object

Access: Read/Write

Prerequisites: None

Returns or sets the coordinates for the origin of the label.

Usage

Label.**Origin** = VdOrigin

Arguments

None

Return Values

VdOrigin. The return/set type for this property. This is of the form VdOrigin Enum.

Parent Property (Label Object)

Scope: Schematic editor

Object: Label Object

Access: Read-Only

Prerequisites: None

Returns the parent object for the label (the object on which the label is placed).

Usage

Label.**Parent**

Arguments

None

Return Values

Object. The object type of the object on which the label is placed.

ResolvedName Property (Label Object)

Scope: Schematic editor

Object: Label Object

Access: Read-Only

Prerequisites: None

Returns the text string of the label and fully resolves any nicknames.

Usage

Label.**ResolvedName**

Arguments

None

Return Values

String. A string containing the text value of the label.

Scope Property (Label Object)

Object: Label Object

Access: Read/Write

Prerequisites: None

Returns or sets the scope of the label.

Usage

```
Label.Scope = VdScope
```

Arguments

None

Return Values

VdScope. The return/set type for this property expressed as VdScope Enum.

Description

This method is used most often on net labels. The scoping can be either local, where the label has visibility on the local block, or global where the label has the scope of the entire design.

Selected Property (Label Object)

Scope: Schematic editor

Object: Label Object

Access: Write-Only

Prerequisites: None

Sets the selection status for the label.

Usage

Label.**Selected** = True | False

Arguments

None

Return Values

True | False. The set type for this property. True - Selects the label. False - Deselects the label.

Sense Property (Label Object)

Scope: Schematic editor

Object: Label Object

Access: Read/Write

Prerequisites: None

Returns or sets the sense of the label that specifies if signal is inverted or not inverted. A label drawn with an overbar indicates an inverted signal.

Usage

Label.**Sense** = VdSense

Arguments

None

Return Values

VdSense. The return/set type for this property of the form VdSense Enum.

Size Property (Label Object)

Scope: Schematic editor

Object: Label Object

Access: Read/Write

Prerequisites: None

Returns or sets the size (text height) of the label.

Usage

Label.**Size** = Long

Arguments

None

Return Values

Long. The return/set type for this property.

TextString Property (Label Object)

Scope: Schematic editor

Object: Label Object

Access: Read/Write

Prerequisites: None

Returns or sets the text value of the label.

Usage

Label.**TextString** = String

Arguments

None

Return Values

String. A string that contains the text value of the label.

Type Property (Label Object)

Scope: Schematic editor

Object: Label Object

Access: Read-Only

Prerequisites: None

Returns the type value VDTS_LABEL, indicating that the object is, in fact, a label.

Usage

Label.**Type**

Arguments

None

Return Values

VdObjectType. The return type for this property. This is of the form VdObjectType Enum.

Visible Property (Label Object)

Scope: Schematic editor

Object: Label Object

Access: Read/Write

Prerequisites: None

Returns or sets the visibility of the label.

Usage

Label.**Visible** = VdLabelVisibility

Arguments

None

Return Values

VdLabelVisibility. The return/set type for this property. This is of the form VdLabelVisibility Enum.

Line Object

The Line Object represents a graphical line in a schematic.

The following table lists methods and properties of the Line object with links to the respective reference pages.

Table 21. Line Object Methods and Properties

Method or Property	Description
AddPoint Method (Line Object)	Adds a new endpoint to the line.
GetNumPoints Method (Line Object)	Returns the number of points in the line.
GetObjectColor Method (Line Object)	Gets the color in which the line is drawn.
GetObjectFillColor Method (Line Object)	Gets the fill color for the line.
GetPoint Method (Line Object)	Returns the coordinates of the specified Point Object.
IsColorAutomatic Method (Line Object)	Determines if the line has an automatic color set for it.
IsFillColorAutomatic Method (Line Object)	Determines if the line has been assigned an automatic fill color.
SetAutomaticColor Method (Line Object)	Sets or unsets the color of a line object as the automatic color for lines.
SetAutomaticFillColor Method (Line Object)	Sets or unsets the fill color of a line object as the automatic fill color for lines.
SetObjectColor Method (Line Object)	Sets the color in which the line is drawn.
SetObjectFillColor Method (Line Object)	Sets the fill color for the line.
Application Property (Line Object)	Returns the Application Object.
LineStyle Property (Line Object)	Returns or sets the style for the line.
Parent Property (Line Object)	Returns the parent Attribute Object for the line.
Selected Property (Line Object)	Sets the selection status for the line.
Type Property (Line Object)	Returns the value VDTS_LINE, indicating that the object is a line.

AddPoint Method (Line Object)

Scope: Schematic editor

Object: Line Object

Prerequisites: None

Adds a new endpoint to the line.

Usage

Line.**AddPoint**(ByVal *NewPoint* As Long) As Boolean

Arguments

- NewPoint

  New end point (packed). The point passed in is packed into a long, with low word representing the X coordinate and the high word representing the Y coordinate. For example: MAKELONG(X,Y)

Return Values

As Boolean. True - the point was added. False - the point could not be added.

Description

This method allows you to add a new endpoint to a line. Xpedition Designer lines can be polygons (more than 2 points).

GetNumPoints Method (Line Object)

Scope: Schematic editor

Object: Line Object

Prerequisites: None

Returns the number of points in the line.

Usage

Line.**GetNumPoints**() As Long

Arguments

None

Return Values

This method returns the number of points on the line.

As Long. Because, Xpedition Designer lines may be polygons (more than 2 points), the return value denotes the number of points in the line.

GetObjectColor Method (Line Object)

Scope: Schematic editor

Object: Line Object

Prerequisites: None

Gets the color in which the line is drawn.

Usage

Line.**GetObjectColor**() As IColor

Arguments

None

Return Values

As IColor. See "CColor Object" on page 256.

GetObjectFillColor Method (Line Object)

Scope: Schematic editor

Object: Line Object

Prerequisites: None

Gets the fill color for the line.

Usage

Line.**GetObjectFillColor**() As IColor

Arguments

None

Return Values

As IColor. The CColor Object of the line.

GetPoint Method (Line Object)

Scope: Schematic editor

Object: Line Object

Prerequisites: None

Returns the coordinates of the specified Point Object.

Note:
Line point indexes are 0 based. That is, the first point in a line is point 0, the second is point 1, and so on.

Usage

Line.**GetPoint**(ByVal *PointNumber* As Long) As IVdPoint

Arguments

- PointNumber

  Specifies the line point (0 through *n*) for which to derive the coordinates.

Return Values

As IVdPoint. The Point Object.

Description

A line may contain many points. Use this method to index each point.

Examples

Show all points.

```
For Index=0 To Line.GetNumPoints()-1
    Set Pt=GetPoint(Index)
    MsgBox "X=" & Pt.X & " Y=" & Pt.Y
Next
```

IsColorAutomatic Method (Line Object)

Scope: Schematic editor

Object: Line Object

Prerequisites: None

Determines if the line has an automatic color set for it.

Usage

Line.**IsColorAutomatic**() As Boolean

Arguments

None

Return Values

As Boolean. True - there is an automatic color set for the line. False - no automatic color is set for the line.

For more information, see "SetAutomaticColor Method (Line Object)" on page 413.

IsFillColorAutomatic Method (Line Object)

Scope: Schematic editor

Object: Line Object

Prerequisites: None

Determines if the line has been assigned an automatic fill color.

Usage

Line.**IsFillColorAutomatic**() As Boolean

Arguments

None

Return Values

As Boolean. True - there is an automatic fill color set for the line. False - no automatic color is set for the line.

For more information, see the "SetAutomaticFillColor Method (Line Object)" on page 414.

SetAutomaticColor Method (Line Object)

Scope: Schematic editor

Object: Line Object

Prerequisites: None

Sets or unsets the color of a line object as the automatic color for lines.

Usage

Line.**SetAutomaticColor**(byVal *bAutomatic* as Boolean)

Arguments

- Automatic

  The color that is set as the default for the object.

  If an object has automatic color, as determined by the IsColorAutomatic Method (Line Object), the actual color of the object is the default for this type of object.

SetAutomaticFillColor Method (Line Object)

Scope: Schematic editor

Object: Line Object

Prerequisites: None

Sets or unsets the fill color of a line object as the automatic fill color for lines.

Usage

Line.**SetAutomaticFillColor**(byVal *bAutomatic* as Boolean)

Arguments

- bAutomatic

  True - sets the automatic fill color for the line.

  False - unsets the automatic fill color for the line.

If an object has automatic fill color, as determined by the "IsFillColorAutomatic Method (Line Object)" on page 412, the actual fill color of the object is the default for this type of object.

SetObjectColor Method (Line Object)

Scope: Schematic editor

Object: Line Object

Prerequisites: None

Sets the color in which the line is drawn.

Usage

Line.**SetObjectColor**(ByVal *newColor* as IColor)

Arguments

- newColor

  The color assigned to the line. The new color is assigned as a Color object, as described in "CColor Object" on page 256.

SetObjectFillColor Method (Line Object)

Scope: Schematic editor

Object: Line Object

Prerequisites: None

Sets the fill color for the line.

Usage

Line.**SetObjectFillColor**(ByVal *newColor* as IColor)

Arguments

- newColor

  The new CColor Object of the line.

Application Property (Line Object)

Scope: Schematic editor

Object: Line Object

Access: Read-Only

Prerequisites: None

Returns the Application Object.

Usage

Line.**Application**

Arguments

None

Return Values

IVdApp. The Application Object.

Description

See Application Object for more information.

LineStyle Property (Line Object)

Scope: Schematic editor

Object: Line Object

Access: Read/Write

Prerequisites: None

Returns or sets the style for the line.

Usage

Line.**LineStyle** = VdLineStyle

Arguments

None

Return Values

VdLineStyle. The return/set type for this property. This is of the form VdLineStyle Enum.

Parent Property (Line Object)

Scope: Schematic editor

Object: Line Object

Access: Read-Only

Prerequisites: None

Returns the parent Attribute Object for the line.

Usage

Line.**Parent**

Arguments

None

Return Values

IVdBlock. The parent Block Object for the line.

See Attribute Object for more information.

Selected Property (Line Object)

Scope: Schematic editor

Object: Line Object

Access: Write-Only

Prerequisites: None

Sets the selection status for the line.

Usage

Line.**Selected** = True | False

Arguments

None

Return Values

True | False. The set type for this property. True - selects the line. False - deselects the line.

Type Property (Line Object)

Scope: Schematic editor

Object: Line Object

Access: Read-Only

Prerequisites: None

Returns the value VDTS_LINE, indicating that the object is a line.

Usage

Line.**Type**

Arguments

None

Return Values

VdObjectType. The return type for this property. This is of the form VdObjectType Enum.

Net Object

This object represents a net or bus on a schematic.

The following table lists methods and properties of the Line object with links to the respective reference pages.

Table 22. Net Object Methods and Properties

Method or Property	Description
AddAttribute Method (Net Object)	Adds an Attribute Object to the net on the specified segment at the coordinates passed.
AddLabel Method (Net Object)	Adds a Label Object to the net.
Connections Method (Net Object)	Returns a collection of Connection Objects. Use this collection to access component pin, net and segment objects.
FindAttribute Method (Net Object)	Locates the attribute specified by the passed argument.
GetConnectedLabel Method (Net Object)	Gets the net label for the specified Segment Object, or any connected segments. This method is useful for identifying segments in a bus net.
GetConnectedNetName Method (Net Object)	Gets the net label for the specified Segment Object, or any connected segments. This method is useful to identify segments in a bus.
GetLabel Method (Net Object)	Gets the net label on the specified Segment Object.
GetObjectColor Method (Net Object)	Gets the color in which the net is drawn.
GetRippers Method (Net Object)	Returns the collection of rippers that are connected to a bus.
GetSegments Method (Net Object)	Returns a collection of the segments that make up a net.
GetSignals Method (Net Object)	Returns a list of constituent bus signals.
GetSingleJointLocs Method (Net Object)	Returns an encoded string of the coordinates of all single joints (dangle boxes).
IsColorAutomatic Method (Net Object)	Determines if the net has an automatic color set for it.
IsSegmentSelected Method (Net Object)	Determines if a net segment is selected.
SelectSegment Method (Net Object)	Selects the specified Segment Object.
SelectSegmentByJointLoc Method (Net Object)	Selects a Segment Object based on the location and type of the lower joint in the segment.
SetAutomaticColor Method (Net Object)	Sets or unsets the color of a net object as the automatic color for nets.
SetObjectColor Method (Net Object)	Sets the color in which the net is drawn.
Application Property (Net Object)	Returns the Application Object.
Attributes Property (Net Object)	Returns a collection object which contains all attributes on this net whether they are visible or not.

Table 22. Net Object Methods and Properties (continued)

Method or Property	Description
Id Property (Net Object)	Returns a unique identifying integer for the net.
LineStyle Property (Net Object)	Returns or sets the line style for the net.
Parent Property (Net Object)	Returns the parent Block Object in which this net was created.
Selected Property (Net Object)	Sets the selection status for the net.
Type Property (Net Object)	Returns VDTS_NET, indicating that the object is, in fact, a net.
UID Property (Net Object)	Returns a Unique Identifying String (UID) for the net.

AddAttribute Method (Net Object)

Scope: Schematic editor

Object: Net Object

Prerequisites: None

Adds an Attribute Object to the net on the specified segment at the coordinates passed.

Usage

Net.**AddAttribute**(ByVal *Segment* As IVdSegment, ByVal *String* As String, ByVal *X* As Long, ByVal *Y* As Long, ByVal *Visibility* As VdVisibilityFlag) As IVdAttr

Arguments

- Segment

  Segment Object with which the attribute is associated.
- String

  Attribute string of the form NAME=VALUE.
- X

  X coordinate of the attribute.
- Y

  Y coordinate of the attribute.
- Visibility

  Specifies the visibility of the attribute in the form VdVisibilityFlag Enum.

Return Values

As IVdAttr. The Attribute Object.

AddLabel Method (Net Object)

Scope: Schematic editor

Object: Net Object

Prerequisites: None

Adds a Label Object to the net.

Usage

Net.**AddLabel**(ByVal *Segment* As IVdSegment, ByVal *String* As String, ByVal *X* As Long, ByVal *Y* As Long) As IVdLabel

Arguments

- Segment

  Segment Object with which the label is associated.
- String

  The text string that defines the label.
- X

  X coordinate of the label.
- Y

  Y coordinate of the label.

Return Values

As IVdLabel. The Label Object.

Description

Adds a Label Object located at the coordinates passed and associated with the Segment Object.

Connections Method (Net Object)

Scope: Schematic editor

Object: Net Object

Prerequisites: None

Returns a collection of Connection Objects. Use this collection to access component pin, net and segment objects.

Usage

Net.**Connections**([ByVal *PinNameFilter* As Variant]) As IVdObjs

Arguments

- PinNameFilter

  (Optional) This is a component filter string that may contain the wildcard character (*).

Return Values

As IVdObjs. A collection of ComponentPin Objects, a Net Objects, and a Segment Objects that, together, comprise the connections for the net.

Description

This method provides a way to traverse all the connections of the net. A connection consists of a ComponentPin Object, a Net Object, and a Segment Object.

See Connection Object for more information.

Examples

Traverse all connections.

```
For Each Conn In Net.Connections
    ...
Next
```

Traverse all clock connections.

```
For Each ClkConnection In Net.Connections("CLOCK*")
    ...
Next
```

FindAttribute Method (Net Object)

Scope: Schematic editor

Object: Net Object

Prerequisites: None

Locates the attribute specified by the passed argument.

Usage

Net.**FindAttribute**(ByVal *AttributeName* As String) As IVdAttr

Arguments

- AttributeName

  Specifies the name of the attribute that is the target of the search.

Return Values

As IVdAttr. The return type for this method. It is the Attribute Object if it is located, otherwise it is a NULL string.

GetConnectedLabel Method (Net Object)

Scope: Schematic editor

Object: Net Object

Prerequisites: None

Gets the net label for the specified Segment Object, or any connected segments. This method is useful for identifying segments in a bus net.

Usage

Net.**GetConnectedLabel**(ByVal *Segment* As IVdSegment) As IVdLabel

Arguments

- Segment

  The Segment Object with which the label is associated.

Return Values

As IVdLabel. The return type for this method. This is the Label Object or NULL if none was found.

GetConnectedNetName Method (Net Object)

Scope: Schematic editor

Object: Net Object

Prerequisites: None

Gets the net label for the specified Segment Object, or any connected segments. This method is useful to identify segments in a bus.

Usage

Net.**GetConnectedNetName**(ByVal *Segment* As IVdSegment) As String

Arguments

- Segment

  The Segment Object with which the label is associated.

Return Values

As String. A string containing the net label.

GetLabel Method (Net Object)

Scope: Schematic editor

Object: Net Object

Prerequisites: None

Gets the net label on the specified Segment Object.

Usage

Net.**GetLabel**(ByVal *Segment* As IVdSegment) As IVdLabel

Arguments

- Segment

  The Segment Object with which the label is associated.

Return Values

As IVdLabel. The return type for this method. This is the Label Object or NULL if none was found.

GetObjectColor Method (Net Object)

Scope: Schematic editor

Object: Net Object

Prerequisites: None

Gets the color in which the net is drawn.

Usage

Net.**GetObjectColor**() As IColor

Arguments

None

Return Values

As IColor. See "CColor Object" on page 256.

GetRippers Method (Net Object)

Object: Net Object

Prerequisites: None

Returns the collection of rippers that are connected to a bus.

Note:
Do not use this method for non-bus nets.

Usage

Net.**GetRippers**() As IVdObjs

Arguments

None

Return Values

As IVdObjs. A collection of Ripper Objects.

GetSegments Method (Net Object)

Scope: Schematic editor

Object: Net Object

Prerequisites: None

Returns a collection of the segments that make up a net.

Usage

Net.**GetSegments**() As IVdObjs

Arguments

None

Return Values

As IVdObjs. A collection of Segment Objects that, together, comprise the net.

GetSignals Method (Net Object)

Object: Net Object

Prerequisites: None

Returns a list of constituent bus signals.

Usage

Net.**GetSignals**() As IStringList

Arguments

None

Return Values

As IStringList. The StringList Collection representing the constituent signal names for the net.

If the net is not a bus, the returned list contains only one element (the name of that net).

GetSingleJointLocs Method (Net Object)

Scope: Schematic editor

Object: Net Object

Prerequisites: None

Returns an encoded string of the coordinates of all single joints (dangle boxes).

Usage

Net.**GetSingleJointLocs**() As String

Arguments

None

Return Values

As String. The encoded string of space-separated X,Y pairs for all joints on the net of type VdJointType.Single.

For more information, see VdJointType Enum.

IsColorAutomatic Method (Net Object)

Scope: Schematic editor

Object: Net Object

Prerequisites: None

Determines if the net has an automatic color set for it.

Usage

Net.**IsColorAutomatic**() As Boolean

Arguments

None

Return Values

As Boolean. True - there is an automatic color set for the net. False - no automatic color is set for the net.

For more information, see “SetAutomaticColor Method (Net Object)” on page 440.

IsSegmentSelected Method (Net Object)

Scope: Schematic editor

Object: Net Object

Prerequisites: None

Determines if a net segment is selected.

Usage

Net.**IsSegmentSelected**(ByVal *Segment* As IVdSegment) As Boolean

Arguments

- Segment

  This argument specifies the Segment Object that is checked.

Return Values

As Boolean. The return type for this method. True - the segment is selected. False - the segment is not selected.

SelectSegment Method (Net Object)

Scope: Schematic editor

Object: Net Object

Prerequisites: None

Selects the specified Segment Object.

Usage

Net.**SelectSegment**(ByVal *Segment* As IVdSegment)

Arguments

- Segment

  Specifies the segment to be selected (see Segment Object for more information).

SelectSegmentByJointLoc Method (Net Object)

Scope: Schematic editor

Object: Net Object

Prerequisites: None

Selects a Segment Object based on the location and type of the lower joint in the segment.

Usage

Net.**SelectSegmentByJointLoc**(ByVal *XCoordinate* As Long, ByVal *YCoordinate* As Long, ByVal *JointType* As VdJointType)

Arguments

Note:
All coordinates are measured in 100ths of an inch.

- XCoordinate

  X coordinate of the low joint.
- YCoordinate

  Y coordinate of the low joint.
- JointType

  This argument specifies the joint type, as described in VdJointType Enum.

Description

A segment joint is defined as being the lower joint if it is closer to the location X=0, Y=0. Two segments can have the same lower joint location and the same joint type at that location. This is a rare condition usually resulting from pasting two copies on top of each other. In this case, only the first occurrence in the Xpedition Designer display list is selected.

See Segment Object for more information.

SetAutomaticColor Method (Net Object)

Scope: Schematic editor

Object: Net Object

Prerequisites: None

Sets or unsets the color of a net object as the automatic color for nets.

Usage

Net.**SetAutomaticColor**(byVal *bAutomatic* as Boolean)

Arguments

- bAutomatic

  The color that is set as the default for the object.

  If an object has automatic color, as determined by the IsColorAutomatic Method (Net Object), the actual color of the object is the default for this type of object.

SetObjectColor Method (Net Object)

Scope: Schematic editor

Object: Net Object

Prerequisites: None

Sets the color in which the net is drawn.

Usage

Net.**SetObjectColor**(ByVal *newColor* as IColor)

Arguments

- newColor

  The color assigned to the net. The new color is assigned as a Color object, as described in "CColor Object" on page 256.

Application Property (Net Object)

Scope: Schematic editor

Object: Net Object

Access: Read-Only

Prerequisites: None

Returns the Application Object.

Usage

Net.**Application**

Arguments

None

Return Values

IVdApp. The Application Object.

Description

See Application Object for more information.

Attributes Property (Net Object)

Scope: Schematic editor

Object: Net Object

Access: Read-Only

Prerequisites: None

Returns a collection object which contains all attributes on this net whether they are visible or not.

Usage

Net.**Attributes**

Arguments

None

Return Values

IVdObjs. The collection of attributes associated with the net.

Description

Nets may have attributes associated with them. A net attribute must be associated with a Segment Object as well.

Id Property (Net Object)

Scope: Schematic editor

Object: Net Object

Access: Read-Only

Prerequisites: None

Returns a unique identifying integer for the net.

Usage

Net.**Id**

Arguments

None

Return Values

Long. A long containing a the unique identifier for the net.

LineStyle Property (Net Object)

Scope: Schematic editor

Object: Net Object

Access: Read/Write

Prerequisites: None

Returns or sets the line style for the net.

Note:
This property only affects buses with a width of 1 and has no effect on buses with widths greater than 1.

Usage

Net.**LineStyle** = VdLineStyle

Arguments

None

Return Values

VdLineStyle. The return/set type for this property. This is of the form VdLineStyle Enum.

Parent Property (Net Object)

Scope: Schematic editor

Object: Net Object

Access: Read-Only

Prerequisites: None

Returns the parent Block Object in which this net was created.

Usage

Net.**Parent**

Arguments

None

Return Values

IVdBlock. The parent Block Object.

Selected Property (Net Object)

Scope: Schematic editor

Object: Net Object

Access: Write-Only

Prerequisites: None

Sets the selection status for the net.

Usage

Net.**Selected** = True | False

Arguments

None

Return Values

True | False. The set type for this property. True - selects the net. False - unselects the net.

Type Property (Net Object)

Scope: Schematic editor

Object: Net Object

Access: Read-Only

Prerequisites: None

Returns VDTS_NET, indicating that the object is, in fact, a net.

Usage

Net.**Type**

Arguments

None

Return Values

VdObjectType. The return type for this property. The value is VDTS_NET, as described in VdObjectType Enum.

UID Property (Net Object)

Scope: Schematic editor

Object: Net Object

Access: Read-Only

Prerequisites: None

Returns a Unique Identifying String (UID) for the net.

Usage

Net.**UID**

Arguments

None

Return Values

String. A string containing the UID for the net in the form $<*sheet number*>N<*ID number*> (for example, $1N34).

PDBPartitions Object

This object allows the user to manipulate PDB partitions data in a Xpedition Designer project file.

The following table lists methods of the PDBPartitions object with links to the respective reference pages.

Table 23. PDBPartions Object Methods

Methods	Description
AppendPDBPartition Method (PDBPartitions Object)	Adds a new entry to the end of the PDB partitions list for a given design.
GetPDBPartition Method (PDBPartitions Object)	Returns the name of the PDB partition.
GetPDBPartitionsArray Method (PDBPartitions Object)	Returns the list of PDB partitions for a given design.
InsertPDBPartition Method (PDBPartitions Object)	Inserts a new entry into the PDB partitions list at the specified index position for a given design.
PDBPartitionExists Method (PDBPartitions Object)	Checks to see if a PDB partition with a specific name exists in the PDB partition collection of a given design.
RemovePDBPartitionByIndex Method (PDBPartitions Object)	Removes the PDB Partition located at the specified index position from the PDB partitions list of a given design.
RemovePDBPartitionByName Method (PDBPartitions Object)	Removes the PDB Partition with the specified name from the PDB partitions list of a given design.

AppendPDBPartition Method (PDBPartitions Object)

Scope: Schematic editor

Object: PDBPartitions Object

Prerequisites: None

Adds a new entry to the end of the PDB partitions list for a given design.

Usage

PDBPartitions.**AppendPDBPartition**(ByVal *sPartition* As String, *siCDBDesing* As String)

Arguments

- sPartition

  PDB partition to add.
- siCDBDesign

  The name of the design.

GetPDBPartition Method (PDBPartitions Object)

Scope: Schematic editor

Object: PDBPartitions Object

Prerequisites: None

Returns the name of the PDB partition.

Usage

PDBPartitions.**GetPDBPartition**(ByVal *Index* As Long, ByVal *siCDBDesign* As String) As String

Arguments

- Index

  An index of PDB partitions in a list. Indexing starts at 1.
- siCDBDesign

  The name of the design.

Return Values

As String. A string containing the name of the PDB partition.

GetPDBPartitionsArray Method (PDBPartitions Object)

Scope: Schematic editor

Object: PDBPartitions Object

Prerequisites: None

Returns the list of PDB partitions for a given design.

Usage

PDBPartitions.**GetPDBPartitionsArray**(ByVal *siCDBDesign* As String) As IStringList

Arguments

- siCDBDesign

  The name of the design.

Return Values

As IStringList. A StringList Collection of PDB partitions.

InsertPDBPartition Method (PDBPartitions Object)

Scope: Schematic editor

Object: PDBPartitions Object

Prerequisites: None

Inserts a new entry into the PDB partitions list at the specified index position for a given design.

Usage

PDBPartitions.**InsertPDBPartition**(ByVal *sPartition* As String, ByVal *Index* As Long, *siCDBDesign* As String)

Arguments

- sPartition

  The name of the PDB partition to add.

- Index

  Index position in the PDB partition list at which to add the new entry. Indexing starts at 1.

  siCDBDesign

  The name of the design.

PDBPartitionExists Method (PDBPartitions Object)

Object: PDBPartitions Object

Prerequisites: None

Checks to see if a PDB partition with a specific name exists in the PDB partition collection of a given design.

Usage

PDBPartitions.**PDBPartitionExists**(ByVal *sPartition* As String, ByVal *siCDBDesign* As String) As Boolean

Arguments

- sPartition

  The PDB partition for which the method checks the existence.
- siCDBDesign

  The name of the design.

Return Values

As Boolean. True - the PDB partition exists in the list. False - the PDB partition does not exist in the list.

RemovePDBPartitionByIndex Method (PDBPartitions Object)

Scope: Schematic editor

Object: PDBPartitions Object

Prerequisites: None

Removes the PDB Partition located at the specified index position from the PDB partitions list of a given design.

Usage

PDBPartitions.**RemovePDBPartitionByIndex**(ByVal *Index* As Long, *siCDBDesign* As String)

Arguments

- Index

  Index position of the partition to be removed from the PDB partition. Indexing starts at 1.
- siCDBDesign

  The name of the design.

RemovePDBPartitionByName Method (PDBPartitions Object)

Scope: Schematic editor

Object: PDBPartitions Object

Prerequisites: None

Removes the PDB Partition with the specified name from the PDB partitions list of a given design.

Usage

PDBPartitions.**RemovePDBPartitionByName**(ByVal *sPartition* As String, *siCDBDesign* As String)

Arguments

- sPartition

  The name of the PDB partition to remove.
- siCDBDesign

  The name of the design.

Pin Object

This object represents a symbol pin on a schematic.

The following table lists methods and properties of the Pin object with links to the respective reference pages.

Table 24. Pin Object Methods and Properties

Method or Property	Description
AddAttribute Method (Pin Object)	Adds an unattached attribute to the pin.
FindAttribute Method (Pin Object)	Finds an attribute on a pin.
GetLocation Method (Pin Object)	Returns the location of the pin.
GetName Method (Pin Object)	Returns the name of the pin.
GetObjectColor Method (Pin Object)	Gets the color in which the pin is drawn.
SetLocation Method (Pin Object)	Specifies the interior and exterior coordinates of the pin.
Application Property (Pin Object)	Returns the Application Object.
Attributes Property (Pin Object)	Returns a collection object which contains all attributes on this pin whether they are visible or not.
Id Property (Pin Object)	Returns a unique identifying integer for the pin.
Label Property (Pin Object)	Returns the Label Object associated with the pin.
Parent Property (Pin Object)	Returns the parent Block Object for this pin.
Selected Property (Pin Object)	Sets the selection status for the pin.
Sense Property (Pin Object)	Returns the sense of the pin, which specifies if signal is inverted or not inverted. A label drawn with an overbar indicates an inverted signal.
Side Property (Pin Object)	Returns the side of the component on which the pin exists.
Type Property (Pin Object)	Returns VDTS_PIN, indicating that the object is, in fact, a pin.
UID Property (Pin Object)	Returns a unique identifying string (UID) for this pin.

AddAttribute Method (Pin Object)

Scope: Schematic editor

Object: Pin Object

Prerequisites: None

Adds an unattached attribute to the pin.

Usage

Pin.**AddAttribute**(ByVal *String* As String, ByVal *X* As Long, ByVal *Y* As Long, ByVal *Visibility* As VdVisibilityFlag) As IVdAttr

Arguments

- String

  String in the form NAME=VALUE.
- X

  The X coordinate of the attribute.
- Y

  The Y coordinate of the attribute.
- Visibility

  The visibility of the attribute in the form VdVisibilityFlag Enum.

Return Values

As IVdAttr. The newly added Attribute Object.

FindAttribute Method (Pin Object)

Scope: Schematic editor

Object: Pin Object

Prerequisites: None

Finds an attribute on a pin.

Usage

Pin.**FindAttribute**(ByVal *AttributeName* As String) As IVdAttr

Arguments

- AttributeName

  The name of the attribute that is the target of the search.

Return Values

As IVdAttr. The return type for this method. The value is the Attribute Object, or NULL, if no attribute is found.

GetLocation Method (Pin Object)

Scope: Schematic editor

Object: Pin Object

Prerequisites: None

Returns the location of the pin.

Note:
All coordinates are measured in 100ths of an inch.

Usage

Pin.**GetLocation**(ByVal *Flag* As VdPinEndType) As IVdPoint

Arguments

- Flag

  Specifies the pin for which to return the coordinates. This is of the form VdPinEndType Enum.

Return Values

As IVdPoint. The Point Object containing the X, Y location of the pin specified by the *Flag* argument.

GetName Method (Pin Object)

Scope: Schematic editor

Object: Pin Object

Prerequisites: None

Returns the name of the pin.

Usage

Pin.**GetName**(ByVal *Flag* As VdNameType) As String

Arguments

- Flag

  This is of the form VdNameType Enum.

Note:
Regardless of the value of this argument, the method returns only the name of the pin, not the hierarchical path.

Return Values

As String. A string containing the name of the pin.

Examples

```
For Each objComp in app.ActiveView.Query(VDM_COMP, VD_ALL)

For Each objConn in objComp.GetConnections

Set objPin = objConn.CompPin.Pin
 objName = objPin.GetName(SHORT_NAME)
 app.AppendOutput("COM Automation","GetName No1=" + CStr(objName))
 objName = objPin.GetName(FULL_PATH_NAME)
 app.AppendOutput("COM Automation","GetName No2=" + CStr(objName))
 objName = objPin.GetName(FULL_PATH_FROM_BLOCK)
 app.AppendOutput("COM Automation","GetName No3=" + CStr(objName))
 Exit For

Next

Next
```

GetObjectColor Method (Pin Object)

Scope: Schematic editor

Object: Pin Object

Prerequisites: None

Gets the color in which the pin is drawn.

Usage

Pin.**GetObjectColor**() As IColor

Arguments

None

Return Values

As Color. See "CColor Object" on page 256.

SetLocation Method (Pin Object)

Scope: Schematic editor

Object: Pin Object

Prerequisites: None

Specifies the interior and exterior coordinates of the pin.

Usage

Pin.**SetLocation**(ByVal *XInterior* As Long, ByVal *YInterior* As Long, ByVal *XBoundary* As Long, ByVal *YBoundary* As Long)

Arguments

Note:
A symbol pin is made up of two points known as the interior and boundary locations. The pin that connects to nets must always be located at the bounding box and thus are known as the boundary coordinates.

All coordinates are measured in 100ths of an inch.

- XInterior

  X coordinate for the pin's interior location.
- YInterior

  Y coordinate for the pin's interior location.
- XBoundary

  X coordinate for the pin's boundary location.
- YBoundary

  Y coordinate for the pin's boundary location.

Application Property (Pin Object)

Scope: Schematic editor

Object: Pin Object

Access: Read-Only

Prerequisites: None

Returns the Application Object.

Usage

Pin.**Application**

Arguments

None

Return Values

IVdApp. The Application Object.

Description

See Application Object for more information.

Attributes Property (Pin Object)

Scope: Schematic editor

Object: Pin Object

Access: Read-Only

Prerequisites: None

Returns a collection object which contains all attributes on this pin whether they are visible or not.

Usage

Pin.**Attributes**

Arguments

None

Return Values

IVdObjs. A collection of the attributes associated with the pin.

Id Property (Pin Object)

Scope: Schematic editor

Object: Pin Object

Access: Read-Only

Prerequisites: None

Returns a unique identifying integer for the pin.

Usage

Pin.**Id**

Arguments

None

Return Values

Long. A value representing the unique ID of the pin.

Label Property (Pin Object)

Scope: Schematic editor

Object: Pin Object

Access: Read-Only

Prerequisites: None

Returns the Label Object associated with the pin.

Usage

Pin.**Label**

Arguments

None

Return Values

IVdLabel. The Label Object.

Parent Property (Pin Object)

Scope: Schematic editor

Object: Pin Object

Access: Read-Only

Prerequisites: None

Returns the parent Block Object for this pin.

Usage

Pin.**Parent**

Arguments

None

Return Values

IVdBlock. The parent Block Object.

Selected Property (Pin Object)

Scope: Schematic editor

Object: Pin Object

Access: Read-Only

Prerequisites: None

Sets the selection status for the pin.

Usage

Pin.**Selected** = True | False

Arguments

None

Return Values

True | False. True - selects the pin. False - deselects the pin.

Sense Property (Pin Object)

Scope: Schematic editor

Object: Pin Object

Access: Read/Write

Prerequisites: None

Returns the sense of the pin, which specifies if signal is inverted or not inverted. A label drawn with an overbar indicates an inverted signal.

CAUTION:
Do not modify the value of this property. By modifying the value, you are editing the pin symbol, which results in an error.

Usage

Pin.**Sense** = VdSense

Arguments

None

Return Values

VdSense. The return/set type for this property. This is of the form VdSense Enum.

Side Property (Pin Object)

Scope: Schematic editor

Object: Pin Object

Access: Read-Only

Prerequisites: None

Returns the side of the component on which the pin exists.

Usage

Pin.**Side**

Arguments

None

Return Values

VdSide. The return type for this property. This is of the form VdSide Enum.

Type Property (Pin Object)

Scope: Schematic editor

Object: Pin Object

Access: Read-Only

Prerequisites: None

Returns VDTS_PIN, indicating that the object is, in fact, a pin.

Usage

Pin.**Type**

Arguments

None

Return Values

VdObjectType. The return type for this property. This is of the form VdObjectType Enum.

UID Property (Pin Object)

Scope: Schematic editor

Object: Pin Object

Access: Read-Only

Prerequisites: None

Returns a unique identifying string (UID) for this pin.

Usage

Pin.**UID**

Arguments

None

Return Values

String. A string containing the UID for the pin in the form $<*sheet number*>P<*ID number*> (for example, $1P34).

Point Object

This object represents a point on a schematic specified by its X,Y coordinates.

The following table lists properties of the Point object with links to the respective reference pages:

Table 25. Point Object Properties

Property	Description
X Property (Point Object)	Returns or sets the X coordinate of a point object.
Y Property (Point Object)	Returns or sets the Y coordinate for a point object.

X Property (Point Object)

Scope: Schematic editor

Object: Point Object

Access: Read/Write

Prerequisites: None

Returns or sets the X coordinate of a point object.

Usage

Point.**X** = Long

Arguments

None

Return Values

Long. A value representing the X coordinate of the point.

Y Property (Point Object)

Scope: Schematic editor

Object: Point Object

Access: Read/Write

Prerequisites: None

Returns or sets the Y coordinate for a point object.

Usage

Point.**Y** = Long

Arguments

None

Return Values

Long. A value representing the Ycoordinate of the point.

ProjectData Object

ProjectData is an automation object which allows managing data stored in Xpedition Designer project file and provides some basic file-related information.

Note:
With the exception of UpdateOtherObjects (which also modifies the database), ProjectData methods that modify project data ("modifier-methods") actually modify only the project file, so Xpedition Designer is not aware of these changes until it reads this project file. Situations where project file contents are unsynchronized with the application's internal state could cause unexpected behavior. These methods should be used carefully by advanced users.

The following table lists methods and properties of the ProjectData object with links to the respective reference pages.

Table 26. ProjectData Object Methods and Properties

Method or Property	Description
AddiCDBDesign Method (ProjectData Object)	Adds new design data to a project file.
GetBordersFilePath Method (ProjectData Object)	Returns the path of the border symbols file.
GetBusContentsFilePath Method (ProjectData Object)	Returns the path of the bus contents file.
GetiCDBDesignRootBlock Method (ProjectData Object)	Returns the name of the top-level block for the given design.
GetiCDBDesigns Method (ProjectData Object)	Returns a collection of design names specified in the project file.
GetiCDBDesignType Method (ProjectData Object)	Returns the type of specified design.
GetiCDBDiscardFilePath Method (ProjectData Object)	Returns the path of the discard file.
GetPCBDesignPath Method (ProjectData Object)	Returns the path of the PCB file for a given design.
GetPDBPartitions Method (ProjectData Object)	Returns the PDBPartitions Object for the project.
GetPinComponentsFilePath Method (ProjectData Object)	Returns the path of the pin components file.
GetProjectFilePath Method (ProjectData Object)	Returns the full file path of project file including project file name.
GetProjectName Method (ProjectData Object)	Returns the project file name, without extension.
GetProjectPath Method (ProjectData Object)	Returns the full file path of the project file, excluding the project file name.
GetSearchPathScheme Method (ProjectData Object)	Returns the library search path scheme for a given design.
GetSymbolPartitions Method (ProjectData Object)	Returns the SymbolPartitions Object for the project.

Table 26. ProjectData Object Methods and Properties (continued)

Method or Property	Description
RemoveiCDBDesign Method (ProjectData Object)	Removes design data from a project file.
RenameiCDBDesign Method (ProjectData Object)	Changes the name for existing design data in the project file.
SetBordersFilePath Method (ProjectData Object)	Sets the path for the borders symbol file.
SetBusContentsFilePath Method (ProjectData Object)	Sets the path of the bus contents file.
SetiCDBDesignRootBlock Method (ProjectData Object)	Sets the name of the top-level block for a given design.
SetiCDBDesignType Method (ProjectData Object)	Sets the type for the specified design.
SetiCDBDiscardFilePath Method (ProjectData Object)	Sets the path for the discard file.
SetPCBDesignPath Method (ProjectData Object)	Sets the path of the PCB file for a given design.
SetPinComponentsFilePath Method (ProjectData Object)	Sets the path for the pin components file.
SetSearchPathScheme Method (ProjectData Object)	Sets the library search path scheme for a given design.
UpdateOtherObjects Method (ProjectData Object)	Updates the specified objects for the project.
CentralLibraryPath Property (ProjectData Object)	Gets or sets the central library path.
iCDBDir Property (ProjectData Object)	Gets the iCDB database path.

AddiCDBDesign Method (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Prerequisites: None

Adds new design data to a project file.

Usage

ProjectData.**AddiCDBDesign**(ByVal *sTopBlockName* As String) As Boolean

Arguments

- sTopBlockName

  The top-level block name for the design.

Return Values

As Boolean. True - the design was successfully added. False - the design could not be added.

GetBordersFilePath Method (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Prerequisites: None

Returns the path of the border symbols file.

Usage

ProjectData.**GetBordersFilePath**(ByVal *resolveSoftPrefix* As Boolean) As String

Arguments

- resolveSoftPrefix

  This argument determines if soft prefixes must be replaced by explicit file path values. The value of this argument can be either True or False.

Return Values

As String. A string containing the path to the border symbols file.

GetBusContentsFilePath Method (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Prerequisites: None

Returns the path of the bus contents file.

Usage

ProjectData.**GetBusContentsFilePath**(ByVal *resolveSoftPrefix* As Boolean) As String

Arguments

- resolveSoftPrefix

  This argument determines if soft prefixes must be replaced by explicit file path values. The value of this argument can be either True or False.

Return Values

As String. A string containing the path to the bus contents file

GetiCDBDesignRootBlock Method (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Prerequisites: None

Returns the name of the top-level block for the given design.

Usage

ProjectData.**GetiCDBDesignRootBlock**(ByVal *siCDBDesign* As String) As String

Arguments

- siCDBDesign

  Design name.

Return Values

As String. A string containing the name of the top-level block.

GetiCDBDesigns Method (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Prerequisites: None

Returns a collection of design names specified in the project file.

Usage

ProjectData.**GetiCDBDesigns**() As IStringList

Arguments

None

Return Values

As StringList. A StringList Collection containing the design names specified in the project file.

GetiCDBDesignType Method (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Prerequisites: None

Returns the type of specified design.

Usage

ProjectData.**GetiCDBDesignType**(ByVal *siCDBDesign* As String) As String

Arguments

- siCDBDesign

  The design name.

Return Values

As String. A string containing the type of the design.

GetiCDBDiscardFilePath Method (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Prerequisites: None

Returns the path of the discard file.

Usage

ProjectData.**GetiCDBDiscardFilePath**(ByVal *resolveSoftPrefix* As Boolean) As String

Arguments

- resolveSoftPrefix

  This argument determines if soft prefixes must be replaced by explicit file path values. The value of this argument can be either True or False.

Return Values

As String. A string containing the path to the discard file.

GetPCBDesignPath Method (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Prerequisites: None

Returns the path of the PCB file for a given design.

Usage

ProjectData.**GetPCBDesignPath**(ByVal *siCDBDesign* As String, ByVal *resolveSoftPrefix* As Boolean) As String

Arguments

- siCDBDesign

  The design name.

- resolveSoftPrefix

  This argument determines if soft prefixes must be replaced by explicit file path values. The value of this argument can be either True or False.

Return Values

As String. A string containing the path to the PCB file.

GetPDBPartitions Method (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Prerequisites: None

Returns the PDBPartitions Object for the project.

Usage

ProjectData.**GetPDBPartitions**() As PDBPartitions

Arguments

None

Return Values

As PDBPartitions. The PDBPartitions Object.

GetPinComponentsFilePath Method (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Prerequisites: None

Returns the path of the pin components file.

Usage

ProjectData.**GetPinComponentsFilePath**(ByVal *resolveSoftPrefix* As Boolean) As String

Arguments

- resolveSoftPrefix

  This argument determines if soft prefixes must be replaced by explicit file path values. The value of this argument is either True or False.

Return Values

As String. A string containing the path to the pin components file.

GetProjectFilePath Method (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Prerequisites: None

Returns the full file path of project file including project file name.

Usage

ProjectData.**GetProjectFilePath**() As String

Arguments

None

Return Values

As String. A string containing the full file path of project file including project file name.

GetProjectName Method (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Prerequisites: None

Returns the project file name, without extension.

Usage

ProjectData.**GetProjectName**() As String

Arguments

None

Return Values

As String. A string containing the project file name.

GetProjectPath Method (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Prerequisites: None

Returns the full file path of the project file, excluding the project file name.

Usage

ProjectData.**GetProjectPath**() As String

Arguments

None

Return Values

As String. A string containing the path to the project file.

GetSearchPathScheme Method (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Prerequisites: None

Returns the library search path scheme for a given design.

Usage

ProjectData.**GetSearchPathScheme**(ByVal *siCDBDesign* As String) As String

Arguments

- siCDBDesign

  The design name.

Return Values

As String. A string containing the library search path scheme.

GetSymbolPartitions Method (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Prerequisites: None

Returns the SymbolPartitions Object for the project.

Usage

ProjectData.**GetSymbolPartitions**() As SymbolPartitions

Arguments

None

Return Values

As SymbolPartitions. The SymbolPartitions Object.

RemoveiCDBDesign Method (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Prerequisites: None

Removes design data from a project file.

Usage

ProjectData.**RemoveiCDBDesign**(ByVal *sTopBlockName* As String) As Boolean

Arguments

- sTopBlockName

  The top-level block name for the design.

Return Values

As Boolean. True - the design was successfully removed. False - the design could not be removed.

RenameiCDBDesign Method (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Prerequisites: None

Changes the name for existing design data in the project file.

Usage

ProjectData.**RenameiCDBDesign**(ByVal *sOldName* As String, ByVal *sNewName* As String) As Boolean

Arguments

- sOldName

  The name of the existing design.
- sNewName

  The new name of the design.

Return Values

As Boolean. True - the design was successfully renamed. False - the design could not be renamed.

SetBordersFilePath Method (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Prerequisites: None

Sets the path for the borders symbol file.

Usage

ProjectData.**SetBordersFilePath**(ByVal *sBorderFilePath* As String)

Arguments

- sBorderFilePath

  A string containing the path to the border symbols file.

SetBusContentsFilePath Method (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Prerequisites: None

Sets the path of the bus contents file.

Usage

ProjectData.**SetBusContentsFilePath**(ByVal *sBusContentsFilePath* As String)

Arguments

- sBusContentsFilePath

  The path of the bus contents file.

SetiCDBDesignRootBlock Method (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Prerequisites: None

Sets the name of the top-level block for a given design.

Usage

ProjectData.**SetiCDBDesignRootBlock** (ByVal *siCDBDesignType* As String, ByVal *siCDBDesign* As String)

Arguments

- siCDBDesignRootBlock

  The name of the block that is to be the root for the design.
- siCDBDesign

  The name of the design.

SetiCDBDesignType Method (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Prerequisites: None

Sets the type for the specified design.

Usage

ProjectData.**SetiCDBDesignType**(ByVal *siCDBDesign* As String, ByVal *siCDBDesignType* As String)

Arguments

- siCDBDesignType

  The target design type.
- siCDBDesign

  The design name.

SetiCDBDiscardFilePath Method (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Prerequisites: None

Sets the path for the discard file.

Usage

ProjectData.**SetiCDBDiscardFilePath**(ByVal *siCDBDiscardFilePath* As String)

Arguments

- siCDBDiscardFilePath

  The path for the discard file.

SetPCBDesignPath Method (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Prerequisites: None

Sets the path of the PCB file for a given design.

Usage

ProjectData.**SetPCBDesignPath**(ByVal *sPCBDesignPath* As String, ByVal *siCDBDesign* As String)

Arguments

- sPCBDesignPath

  The path of the PCB file.
- siCDBDesign

  The design name.

SetPinComponentsFilePath Method (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Prerequisites: None

Sets the path for the pin components file.

Usage

ProjectData.**SetPinComponentsFilePath**(ByVal *sPinComponentsFilePath* As String)

Arguments

- sPinComponentsFilePath

  The path for the pin components file.

SetSearchPathScheme Method (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Prerequisites: None

Sets the library search path scheme for a given design.

Usage

ProjectData.**SetSearchPathScheme**(ByVal *siCDBSearchPathScheme*, ByVal *siCDBDesign* As String)

Arguments

- siCDBSearchPathScheme

  The library search path scheme.
- siCDBDesign

  The name of the design.

UpdateOtherObjects Method (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Prerequisites: None

Updates the specified objects for the project.

CAUTION:
The UpdateOtherObjects method modifies the database as well as the project file.

Usage

ProjectData.**UpdateOtherObjects**(ByVal *eWhatToUpdate* As VdUpdateOtherObjects, ByVal *eUpdateScope* As VdUpdateOOScope)

Arguments

- eWhatToUpdate

  The objects to update (VdUpdateOtherObjects Enum).
- eUpdateScope

  The scope of the update (VdUpdateOOScope Enum).

CentralLibraryPath Property (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Access: Read/Write

Prerequisites: None

Gets or sets the central library path.

Usage

ProjectData.**CentralLibraryPath** = String

Arguments

None

Return Values

String. A string containing the path to the central library.

iCDBDir Property (ProjectData Object)

Scope: Schematic editor

Object: ProjectData Object

Access: Read-Only

Prerequisites: None

Gets the iCDB database path.

Usage

ProjectData.**iCDBDir**

Arguments

None

Return Values

String. A string containing the iCDB database path.

Rect Object

This object represents a graphic rectangle on a schematic, specified by its coordinates.

The following table lists properties of the Rect object with links to the respective reference pages.

Table 27. Rect Object Properties

Property	Description
Bottom Property (Rect Object)	Returns or sets the Y coordinate of the bottom side of the rectangle.
Left Property (Rect Object)	Returns or sets the X coordinate of the left side of the rectangle.
Right Property (Rect Object)	Returns or sets the X coordinate of the right side of the rectangle.
Top Property (Rect Object)	Returns or sets the Y coordinate of the top of the rectangle.

Bottom Property (Rect Object)

Scope: Schematic editor

Object: Rect Object

Access: Read/Write

Prerequisites: None

Returns or sets the Y coordinate of the bottom side of the rectangle.

Usage

Rect.**Bottom** = Long

Arguments

None

Return Values

Long. A value representing the Y coordinate of the bottom side of the rectangle.

Left Property (Rect Object)

Scope: Schematic editor

Object: Rect Object

Access: Read/Write

Prerequisites: None

Returns or sets the X coordinate of the left side of the rectangle.

Usage

Rect.**Left** = Long

Arguments

None

Return Values

Long. A value representing the X coordinate of the left side of the rectangle.

Right Property (Rect Object)

Scope: Schematic editor

Object: Rect Object

Access: Read/Write

Prerequisites: None

Returns or sets the X coordinate of the right side of the rectangle.

Usage

Rect.**Right** = Long

Arguments

None

Return Values

Long. A value representing the X coordinate of the right side of the rectangle.

Top Property (Rect Object)

Scope: Schematic editor

Object: Rect Object

Access: Read/Write

Prerequisites: None

Returns or sets the Y coordinate of the top of the rectangle.

Usage

Rect.**Top** = Long

Arguments

None

Return Values

Long. A value representing the Y coordinate of the top side of the rectangle.

Ripper Object

This object represents a bus ripper on a schematic.

The following table lists methods of the Ripper object with links to the respective reference pages.

Table 28. Ripper Object Methods

Method	Description
GetConnectedObject Method (Ripper Object)	Returns the net or bus that is located at the other side of the bus ripper.
GetConnectedObjects Method (Ripper Object)	Returns the collection of Net Objects that are connected to that bus ripper.
GetMappedSignal Method (Ripper Object)	Returns the name of the signal that is mapped to the given signal by the bus ripper.

GetConnectedObject Method (Ripper Object)

Object: Ripper Object

Prerequisites: None

Returns the net or bus that is located at the other side of the bus ripper.

Usage

Ripper.**GetConnectedObject**(ByVal *Net* As IVdNet) As IVdNet

Arguments

- Net

  A Net Object connected to one side of the ripper.

Return Values

As IVdNet. The Net Object connected to the other side of the ripper.

Description

This method is useful for following connectivity. For example, if you have a reference to a Net Object connected to a bus ripper, this method lets you "walk through" that ripper and obtain a reference to another Net Object connected to this bus ripper.

GetConnectedObjects Method (Ripper Object)

Object: Ripper Object

Prerequisites: None

Returns the collection of Net Objects that are connected to that bus ripper.

Usage

Ripper.**GetConnectedObjects**() As IVdObjs

Arguments

None

Return Values

As IVdObjs. The collection of Net Objects.

GetMappedSignal Method (Ripper Object)

Scope: Schematic editor

Object: Ripper Object

Prerequisites: None

Returns the name of the signal that is mapped to the given signal by the bus ripper.

Usage

Ripper.**GetMappedSignal**(ByVal *Net* as IVdNet, ByVal *sSignal* As String) As String

Arguments

- Net

  The net or bus which includes the given signal.
- sSignal

  The name of the signal.

Return Values

As String. A string containing the name of the signal that is mapped to *sSignal*.

SchematicSheetDocument Object

This object represents one sheet of a schematic.

The following table lists methods and properties of the SchematicSheetDocument object with links to the respective reference pages.

Table 29. SchematicSheetDocument Object Methods and Properties

Method or Property	Description
Activate Method (SchematicSheetDocument Object)	Assigns the first view of the Views collection owned by this document as the active view.
Close Method (SchematicSheetDocument Object)	Closes the document and flushes it from memory.
DiscardSymbolChanges Method (SchematicSheetDocument Object)	Discards local symbol without updating design or saving changes.
ExportMetafile Method (SchematicSheetDocument Object)	Exports the current document as a metafile for plotting.
GetViews Method (SchematicSheetDocument Object)	Returns a collection of View objects which contain this document.
IsReadOnly Method (SchematicSheetDocument Object)	Determines if the current document is read-only (locked).
Print Method (SchematicSheetDocument Object)	Prints a document range to the default printer.
ReRead Method (SchematicSheetDocument Object)	Causes the specified page of the current document to be reread from disk.
Save Method (SchematicSheetDocument Object)	Saves changes made in the embedded symbol editor to a non-local symbol.
SaveAs Method (SchematicSheetDocument Object)	Saves non-local symbol opened in the embedded symbol editor to a different name.
UpdateSymbolInDesign Method (SchematicSheetDocument Object)	Updates symbol in the schematic to include the current local changes.
Application Property (SchematicSheetDocument Object)	Returns the application property for the document.
FullName Property (SchematicSheetDocument Object)	Returns the name of the schematic sheet.

Table 29. SchematicSheetDocument Object Methods and Properties (continued)

Method or Property	Description
Name Property (SchematicSheetDocument Object)	Sets or returns the name of the schematic sheet.
Parent Property (SchematicSheetDocument Object)	Returns the parent Attribute Object for the document.

Activate Method (SchematicSheetDocument Object)

Scope: Schematic editor

Object: SchematicSheetDocument Object

Prerequisites: None

Assigns the first view of the Views collection owned by this document as the active view.

Usage

SchematicSheetDocument.**Activate**()

Arguments

None

Close Method (SchematicSheetDocument Object)

Scope: Schematic editor

Object: SchematicSheetDocument Object

Prerequisites: None

Closes the document and flushes it from memory.

Note:
Arguments for this method are deprecated; however, they are still required. A runtime error will result if you do not provide arguments.

Usage

SchematicSheetDocument.**Close**(ByVal *SaveChanges* As Boolean, ByVal *FileName* As String)

Arguments

- SaveChanges

  (Deprecated) True - save any changes to the schematic before closing. False - close the schematic without saving.

- FileName

  (Deprecated) A string containing the name of the schematic document to close.

DiscardSymbolChanges Method (SchematicSheetDocument Object)

Scope: Schematic editor - Embedded symbol editor (ESE)

Object: SchematicSheetDocument Object

Prerequisites: None

Discards local symbol without updating design or saving changes.

Usage

SchematicSheetDocument.**DiscardSymbolChanges**()

Arguments

None

Examples

```
Set app = CreateObject("Viewdraw.Application")
Scripting.AddTypeLibrary("Viewdraw.Application")
app.OpenProject("PROJECT_NAME.prj")
Set ese_test = app.SchematicSheetDocuments.OpenSymbol("my_loc_sym", "1")
Set ese_view = app.ActiveView
Set ese_symbol = ese_view.activeblock
Call ese_symbol.AddBox(0,0,50,50)
ese_test.DiscardSymbolChanges
```

ExportMetafile Method (SchematicSheetDocument Object)

Scope: Schematic editor

Object: SchematicSheetDocument Object

Prerequisites: None

Exports the current document as a metafile for plotting.

Usage

SchematicSheetDocument.**ExportMetafile**(ByVal *OutputName* As String)

Arguments

- OutputName

  Name of the output file.

GetViews Method (SchematicSheetDocument Object)

Scope: Schematic editor

Object: SchematicSheetDocument Object

Prerequisites: None

Returns a collection of View objects which contain this document.

Usage

SchematicSheetDocument.**GetViews**() As IVdViews

Arguments

None

Return Values

As IVdViews. The View Object collection.

IsReadOnly Method (SchematicSheetDocument Object)

Object: SchematicSheetDocument Object

Prerequisites: None

Determines if the current document is read-only (locked).

Usage

SchematicSheetDocument.**IsReadOnly**() As Boolean

Arguments

None

Return Values

As Boolean. True - the document is read-only. False - the document is not read-only.

Print Method (SchematicSheetDocument Object)

Scope: Schematic editor

Object: SchematicSheetDocument Object

Prerequisites: None

Prints a document range to the default printer.

Usage

SchematicSheetDocument.**Print**(ByVal *From* As Integer, ByVal *To* As Integer, ByVal *Copies* As Integer)

Arguments

- From

  Starting page number.
- To

  Ending page number.
- Copies

  Number of copies.

ReRead Method (SchematicSheetDocument Object)

Scope: Schematic editor

Object: SchematicSheetDocument Object

Prerequisites: None

Causes the specified page of the current document to be reread from disk.

Usage

SchematicSheetDocument.**ReRead**(ByVal *SheetName* As String) As Boolean

Arguments

- SheetName

  Name of the sheet from which to read.

Return Values

As Boolean. The return type for this method. True - Indicates that the page was read. False - Indicates that the page could not be read.

Description

ReRead loads the specified page of the current document from disk. This method throws away any current changes that have not been saved. All objects that are owned by this document, like the views, should be released prior to using this method.

This method returns True if it is successful. The most likely failure is caused by Automation programmers that do not check for the existence of the specified page prior to trying to load it.

Save Method (SchematicSheetDocument Object)

Scope: Schematic editor - Embedded symbol editor (ESE)

Object: SchematicSheetDocument Object

Prerequisites: Symbol opened to edition from Library Manager (non-local symbol)

Saves changes made in the embedded symbol editor to a non-local symbol.

Usage

SchematicSheetDocument.**Save**()

Arguments

None

Examples

```
Set app = CreateObject("Viewdraw.Application")
Scripting.AddTypeLibrary("Viewdraw.Application")

run_ESE="-lmc C:\MentorGraphics\EEVX.2.10\SDD_HOME\standard\examples\"
  + "SampleLib2007\SymbolLibs -partition Discrete -- cap.1"

app.RunISE(run_ESE)

Set myDoc = app.ActiveDocument
Set myView = app.ActiveView
```

```
’Change the symbol
Set ese_symbol = myView.ActiveBlock
Call ese_symbol.AddArc(200,200,250,250,200,300)
```

```
’Save changes in the symbol
myDoc.Save
```

SaveAs Method (SchematicSheetDocument Object)

Scope: Schematic editor - Embedded symbol editor (ESE)

Object: SchematicSheetDocument Object

Prerequisites: Symbol opened to edition from Library Manager (in LM mode) or from Databook (in Netlist mode) - non-local symbol

Saves non-local symbol opened in the embedded symbol editor to a different name.

Usage

SchematicSheetDocument.**SaveAs**(ByVal *Symbol* As String)

Arguments

- Symbol

  A string containing the name of the local symbol.

Examples

```
Set app = CreateObject("Viewdraw.Application")
Scripting.AddTypeLibrary("Viewdraw.Application")

run_ESE="-lmc C:\MentorGraphics\EEVX.2.10\SDD_HOME\standard\examples\"
  + "SampleLib2007\SymbolLibs -partition Discrete -- cap.1"

app.RunISE(run_ESE)

Set myDoc = app.ActiveDocument
Set myView = app.ActiveView
'Change the symbol
Set ese_symbol = myView.ActiveBlock
Call ese_symbol.AddArc(200,200,250,250,200,300)

'Save on disk in Netlist flow
myDoc.saveas "c:\MyLibs\Discrete\sym\newcap.1"

'Save in Central Library in partition Discrete
myDoc.saveas "Discrete:newcap.1"
```

UpdateSymbolInDesign Method (SchematicSheetDocument Object)

Scope: Schematic editor - Embedded symbol editor (ESE)

Object: SchematicSheetDocument Object

Prerequisites: None

Updates symbol in the schematic to include the current local changes.

Usage

SchematicSheetDocument.**UpdateSymbolInDesign**()

Arguments

None

Examples

```
Set app = CreateObject("Viewdraw.Application")
Scripting.AddTypeLibrary("Viewdraw.Application")
app.OpenProject("PROJECT_NAME.prj")
Set ese_test = app.SchematicSheetDocuments.OpenSymbol("my_loc_sym", "1")
Set ese_view = app.ActiveView
Set ese_symbol = ese_view.ActiveBlock
Call ese_symbol.AddBox(0,0,50,50)
ese_test.UpdateSymbolInDesign
```

Application Property (SchematicSheetDocument Object)

Scope: Schematic editor

Object: SchematicSheetDocument Object

Access: Read-Only

Prerequisites: None

Returns the application property for the document.

Usage

SchematicSheetDocument.**Application**

Arguments

None

Return Values

IVdApp. The Application Object.

FullName Property (SchematicSheetDocument Object)

Scope: Schematic editor

Object: SchematicSheetDocument Object

Access: Read-Only

Prerequisites: None

Returns the name of the schematic sheet.

Usage

SchematicSheetDocument.**FullName**

Arguments

None

Return Values

String. The return type for this property.

Description

This property returns the same value as the Name Property (SchematicSheetDocument Object).

Name Property (SchematicSheetDocument Object)

Scope: Schematic editor

Object: SchematicSheetDocument Object

Access: Read/Write

Prerequisites: None

Sets or returns the name of the schematic sheet.

Usage

SchematicSheetDocument.**Name**

Arguments

None

Return Values

String. The return type for this property.

Parent Property (SchematicSheetDocument Object)

Scope: Schematic editor

Object: SchematicSheetDocument Object

Access: Read-Only

Prerequisites: None

Returns the parent Attribute Object for the document.

Usage

SchematicSheetDocument.**Parent**

Arguments

None

Return Values

IVdDocs. The return type for this property.

See Attribute Object for more information.

Segment Object

This object represents a portion of a net or bus on a schematic.

The following table lists methods and properties of the Segment object with links to the respective reference pages.

Table 30. Segment Object Methods and Properties

Method or Property	Description
GetJointType Method (Segment Object)	Returns the type of a specified joint.
IsBus Method (Segment Object)	Determines if the segment is a bus.
Location Method (Segment Object)	Returns the coordinates of specified joint in the form of a Point Object.
Application Property (Segment Object)	Returns the Application Object.
Attributes Property (Segment Object)	Returns a collection object that contains all Attribute Objects on this segment, whether they are visible or not.
Parent Property (Segment Object)	Returns the parent Net Object of the segment.
Type Property (Segment Object)	Returns VDTS_SEGMENT, indicating that the object is, in fact, a segment.

GetJointType Method (Segment Object)

Scope: Schematic editor

Object: Segment Object

Prerequisites: None

Returns the type of a specified joint.

Note:
All coordinates are measured in 100ths of an inch.

Usage

Segment.**GetJointType**(ByVal *WhichJoint* As VdWhichJoint) As VdJointType

Arguments

- WhichJoint

  Specifies the location of the joint for which the type is determined. This is of the form VdWhichJoint Enum.

Return Values

As VdJointType. The joint type.

A segment has two joints known as the High and Low joints. The Low joint always has lesser magnitude values for both X,Y coordinates than the High joint (sorted).

IsBus Method (Segment Object)

Object: Segment Object

Prerequisites: None

Determines if the segment is a bus.

Usage

Segment.**IsBus**() As Boolean

Arguments

None

Return Values

As Boolean. True - the segment is a bus. False - the segment is not a bus.

Location Method (Segment Object)

Scope: Schematic editor

Object: Segment Object

Prerequisites: None

Returns the coordinates of specified joint in the form of a Point Object.

Note:
All coordinates are measured in 100ths of an inch.

Usage

Segment.**Location**(ByVal *WhichJoint* As VdWhichJoint) As IVdPoint

Arguments

- WhichJoint

  A segment has two joints known as the High and Low joints. The Low joint always has lesser magnitude values for both X,Y coordinates than the High joint (sorted). *WhichJoint* specifies the joint of the segment for which the coordinates are returned. This is of the form VdWhichJoint Enum.

Return Values

As IVdPoint. The Point Object.

Application Property (Segment Object)

Scope: Schematic editor

Object: Segment Object

Access: Read-Only

Prerequisites: None

Returns the Application Object.

Usage

Segment.**Application**

Arguments

None

Return Values

IVdApp. The Application Object.

Description

See Application Object for more information.

Attributes Property (Segment Object)

Scope: Schematic editor

Object: Segment Object

Access: Read-Only

Prerequisites: None

Returns a collection object that contains all Attribute Objects on this segment, whether they are visible or not.

Usage

Segment.**Attributes**

Arguments

None

Return Values

IVdObjs. A collection of the Attribute Objects associated with the segment.

Description

See Attribute Object for more information.

Parent Property (Segment Object)

Scope: Schematic editor

Object: Segment Object

Access: Read-Only

Prerequisites: None

Returns the parent Net Object of the segment.

Usage

Segment.**Parent**

Arguments

None

Return Values

IVdNet. The parent Net Object.

Type Property (Segment Object)

Scope: Schematic editor

Object: Segment Object

Access: Read-Only

Prerequisites: None

Returns VDTS_SEGMENT, indicating that the object is, in fact, a segment.

Usage

Segment.**Type**

Arguments

None

Return Values

VdObjectType. The return/set type for this property. This is of the form VdObjectType Enum.

ShellExecution Object

The ShellExecution object defines the characteristics necessary to run an external program in the application.

Note:
The ShellExecution object is in the "Viewlogic.Exec" COM server. The program identification (ProgID) for this COM server is Viewlogic.Exec.

The following tables list methods, properties and events of the ShellExecution object with links to the respective reference pages:

Open Method (ShellExecution Object)
PID Property (ShellExecution Object)
RedirectOutput Property (ShellExecution Object)
Run Method (ShellExecution Object)
Terminate Method (ShellExecution Object)
WorkingDirectory Property (ShellExecution Object)
Exit Event (ShellExecution Object)
Start Event (ShellExecution Object)
StdError Event (ShellExecution Object)
StdOutput Event (ShellExecution Object)

Open Method (ShellExecution Object)

Scope: Schematic editor

Object: ShellExecution Object

Prerequisites: None

Opens a file with the registered viewer.

Usage

Object.**Open** (ByVal *URL* As String) As Integer

Arguments

- URL

  A string that specifies the URL or filepath.

Return Values

Integer.

Examples

This example uses the Open method to open a text file

```
Dim execObj
Set execObj = CreateObject("Viewlogic.Exec")

'Open a text file
Call execObj.Open("file:///C:/temp/Test.txt")
```

You can use either a URL or filepath in the argument:

```
Call execObj.Open("C:\temp\Test.txt")
```

THis example uses the Open method to open a web page

```
Call execObj.Open("https://www.sw.siemens.com/")
```

PID Property (ShellExecution Object)

Scope: Schematic editor

Object: ShellExecution Object

Access: Read/Write

Prerequisites: None

Returns or sets the Process ID.

Usage

Object.**PID** = Integer

Arguments

None

Return Values

Integer. A value representing the Process ID.

RedirectOutput Property (ShellExecution Object)

Scope: Schematic editor

Object: ShellExecution Object

Access: Read/Write

Prerequisites: None

Returns or sets the RedirectOutput flag.

Usage

Object.**RedirectOutput** = True | False

Arguments

None

Return Values

True | False. If True, the RedirectOutput flag is set True. If False, the RedirectOutput flag is set False.

Run Method (ShellExecution Object)

Scope: Schematic editor

Object: ShellExecution Object

Prerequisites: None

Runs a command.

Usage

Object.**Run**(
ByVal *strCommand* As String,
[ByVal *WindowStyle* As Variant],
[ByVal *WaitOnReturn* As Variant]) As Integer

Arguments

- strCommand

  String containing the command to be run, plus any arguments related to the command.
- WindowStyle

  (Optional) Variant that controls visibility of the command window in which this method is executed.

  - A value of 1 (WindowVisible) sets the command window to be visible.
  - A value of 0 (WindowInvisible) hides the command window.

  The default value is 1 (WindowVisible). See WindowStyle Enum topic for more information.
- WaitOnReturn

  (Optional) Boolean that controls whether the system waits for the Run method to finish executing before it proceeds with the rest of the script.

Return Values

Integer

Examples

This example uses 7-Zip to add an existing text file *MyNetReport.txt* to an archive. It also adds the *\data* folder and the files and subfolders contained therein.

```
Dim execObj
Set execObj = CreateObject("Viewlogic.Exec")

'Hide the command window
Dim visibleInt : visibleInt = 0
'Make system wait for Run method to finish
Dim waitBool : waitBool = True
```

```
' Zip the file and the \data folder
Call execObj.Run("C:/Program Files/7-Zip/7z.exe a NetReport.zip" & _
" MyNetReport.txt -r C:/WDIR/EEVX.2.12/tmp/data/", visibleInt, waitBool)
```

Terminate Method (ShellExecution Object)

Scope: Schematic editor

Object: ShellExecution Object

Prerequisites: None

Terminates the command.

Usage

Object.**Terminate**(ByVal *ExitCode* As Integer) As Boolean

Arguments

- ExitCode

  Integer

Return Values

Boolean that returns True if the command termintated successfully. Otherwise, returns False.

Examples

This example terminates a running command.

```
Dim execObj
Set execObj = CreateObject("Viewlogic.Exec")

'Terminate the command
Dim ExitCode : ExitCode = 0
Call execObj.Terminate(ExitCode)
```

WorkingDirectory Property (ShellExecution Object)

Scope: Schematic editor

Object: ShellExecution Object

Access: Read/Write

Prerequisites: None

Returns or sets the default directory in which to Run the program.

Usage

Object.**WorkingDirectory** = String

Arguments

None

Return Values

String. A value that represents the default directory.

Exit Event (ShellExecution Object)

Scope: Schematic editor

Object: ShellExecution Object

Prerequisites: None

Occurs when the program has terminated.

Usage

Sub **ShellExecution_Exit**(ByVal *PID* As Integer, ByVal *ExitCode* As Integer)

Arguments

- PID

  Integer that corresponds to the Process ID.
- ExitCode

  Integer that corresponds to the ExitCode.

Start Event (ShellExecution Object)

Scope: Schematic editor

Object: ShellExecution Object

Prerequisites: None

Occurs when the subprocess has started.

Usage

Sub **ShellExecution_Start**(ByVal *PID* As Integer)

Arguments

- PID

  Integer that corresponds to the Process ID.

StdError Event (ShellExecution Object)

Scope: Schematic editor

Object: ShellExecution Object

Prerequisites: None

Occurs whenever a line of error is produced (stderr).

Usage

Sub **ShellExecution_StdError**(ByVal *PID* As Integer, ByVal *OutputString* As String)

Arguments

- PID

  Integer that corresponds to a Process ID.
- OutputString

  String that corresponds to an error (stderr).

StdOutput Event (ShellExecution Object)

Scope: Schematic editor

Object: ShellExecution Object

Prerequisites: None

Occurs whenever a line of output is produced (stdout).

Usage

Sub **ShellExecution_StdOutput**(ByVal *PID* As Integer, ByVal *OutputString* As String)

Arguments

- PID

  Integer that corresponds to a Process ID.
- OutputString

  String that corresponds to a line of output (stdout).

SymbolPartitions Object

This object allows you to manipulate symbol partitions data in a project file.

The following table lists methods of the SymbolPartitions object with links to the respective reference pages.

Table 31. SymbolPartitions Object Methods

Method	Description
AppendSymbolPartition Method (SymbolPartitions Object)	Adds a new entry to the end of the symbol partitions list for a given design.
GetSymbolPartition Method (SymbolPartitions Object)	Returns the name of a symbol partition (specified by index) for a given design from the symbol partitions list.
GetSymbolPartitionsArray Method (SymbolPartitions Object)	Returns the list of symbol partitions for a given design.
GetSymbolPartitionsCount Method (SymbolPartitions Object)	Returns the number of symbol partitions for a given design.
InsertSymbolPartition Method (SymbolPartitions Object)	Inserts a new entry into the symbol partitions list for a design, at the specified position.
RemoveSymbolPartitionByIndex Method (SymbolPartitions Object)	Removes an entry (specified by index) from the symbol partitions list for a given design.
RemoveSymbolPartitionByName Method (SymbolPartitions Object)	Removes an entry (specified by name) from the symbol partitions list for a given design.
SymbolPartitionExists Method (SymbolPartitions Object)	Indicates whether the specified name exists in the symbol partitions list for a given design.

AppendSymbolPartition Method (SymbolPartitions Object)

Scope: Schematic editor

Object: SymbolPartitions Object

Prerequisites: None

Adds a new entry to the end of the symbol partitions list for a given design.

Usage

SymbolPartitions.**AppendSymbolPartition**(ByVal *sPartition* As String, *siCDBDesign* As String)

Arguments

- sPartition

  The name of the symbol partition to append to the list.
- siCDBDesign

  The name of the design.

GetSymbolPartition Method (SymbolPartitions Object)

Scope: Schematic editor

Object: SymbolPartitions Object

Prerequisites: None

Returns the name of a symbol partition (specified by index) for a given design from the symbol partitions list.

Usage

SymbolPartitions.**GetSymbolPartition**(ByVal *Index* As Long, ByVal *siCDBDesign* As String) As String

Arguments

- Index

  Index of the element in the collection. Indexing starts at 1.
- siCDBDesign

  The design name.

Return Values

As String. A string containing the name of the symbol partition.

GetSymbolPartitionsArray Method (SymbolPartitions Object)

Scope: Schematic editor

Object: SymbolPartitions Object

Prerequisites: None

Returns the list of symbol partitions for a given design.

Usage

*SymbolPartitions.***GetSymbolPartitionsArray***(ByVal siCDBDesign As String) As IStringList*

Arguments

- siCDBDesign

  The design name.

Return Values

As StringList. A StringList Collection of the symbol partitions for the design.

GetSymbolPartitionsCount Method (SymbolPartitions Object)

Scope: Schematic editor

Object: SymbolPartitions Object

Prerequisites: None

Returns the number of symbol partitions for a given design.

Usage

SymbolPartitions.**GetSymbolPartitionsCount**(ByVal *siCDBDesign* As String) As Long

Arguments

- sciCDBDesign

  The design name.

Return Values

As Long. A value representing the number of symbol partitions in the list.

InsertSymbolPartition Method (SymbolPartitions Object)

Scope: Schematic editor

Object: SymbolPartitions Object

Prerequisites: None

Inserts a new entry into the symbol partitions list for a design, at the specified position.

Usage

SymbolPartitions.**InsertSymbolPartition**(ByVal *sPartition* As String, ByVal *Index* As Long, *siCDBDesing* As String)

Arguments

- sPartition

  The name of the symbol partition to add.
- Index

  Index of the element in the collection. Indexing starts at 1.
- siCDBDesign

  The name of the design.

RemoveSymbolPartitionByIndex Method (SymbolPartitions Object)

Scope: Schematic editor

Object: SymbolPartitions Object

Prerequisites: None

Removes an entry (specified by index) from the symbol partitions list for a given design.

Usage

*SymbolPartitions.***RemoveSymbolPartitionByIndex***(ByVal Index As Long, siCDBDesign As String)*

Arguments

- Index

  Index of the element in the collection. Indexing starts at 1.
- siCDBDesign

  The name of the design.

RemoveSymbolPartitionByName Method (SymbolPartitions Object)

Scope: Schematic editor

Object: SymbolPartitions Object

Prerequisites: None

Removes an entry (specified by name) from the symbol partitions list for a given design.

Usage

*SymbolPartitions.***RemoveSymbolPartitionByName***(ByVal sPartition As String, siCDBDesign As String)*

Arguments

- sPartition

  The name of the symbol partition to remove from the list.
- siCDBDesign

  The name of the design.

SymbolPartitionExists Method (SymbolPartitions Object)

Scope: Schematic editor

Object: SymbolPartitions Object

Prerequisites: None

Indicates whether the specified name exists in the symbol partitions list for a given design.

Usage

SymbolPartitions.**SymbolPartitionExists**(ByVal *sPartition* As String, ByVal *siCDBDesign* As String) As Boolean

Arguments

- sPartition

  The name of the symbol partition for which to search.
- siCDBDesign

  The name of the design.

Return Values

As Boolean. True - the symbol partition exists in the list. False - the symbol partition does not exist in the list.

Text Object

This object represents text on a Schematic.

The following table lists methods and properties of the Text object with links to the respective reference pages.

Table 32. Text Object Methods and Properties

Method	Description
GetLocation Method (Text Object)	Returns the coordinates of the text object in the form of a Point Object.
GetObjectColor Method (Text Object)	Gets the color in which the text is drawn.
IsColorAutomatic Method (Text Object)	Determines if the text has an automatic color set for it.
SetAutomaticColor Method (Text Object)	Sets or unsets the color of a text object as the automatic color for text.
SetLocation Method (Text Object)	Specifies the coordinates of the text object.
SetObjectColor Method (Text Object)	Sets the color in which the text is drawn.
Application Property (Text Object)	Returns the Application Object.
Font Property (Text Object)	Returns or sets the font in which the text is written.
Orientation Property (Text Object)	Returns or sets the orientation of the text.
Origin Property (Text Object)	Returns or sets the origin for the text.
Parent Property (Text Object)	Returns the parent Block Object for the text.
Selected Property (Text Object)	Sets the selection status for the text.
Size Property (Text Object)	Returns or sets the size (height) of the text.
TextString Property (Text Object)	Returns or sets the text string; that is, the actual contents of the text.
Type Property (Text Object)	Returns VDTS_TEXT, indicating that the object is, in fact, text.

GetLocation Method (Text Object)

Scope: Schematic editor

Object: Text Object

Prerequisites: None

Returns the coordinates of the text object in the form of a Point Object.

Note:
All coordinates are measured in 100ths of an inch.

Usage

Text.**GetLocation**() As IVdPoint

Arguments

None

Return Values

As IVdPoint. The Point Object denoting the text coordinates.

GetObjectColor Method (Text Object)

Scope: Schematic editor

Object: Text Object

Prerequisites: None

Gets the color in which the text is drawn.

Usage

Text.**GetObjectColor**() As IColor

Arguments

None

Return Values

As Color. See "CColor Object" on page 256.

IsColorAutomatic Method (Text Object)

Scope: Schematic editor

Object: Text Object

Prerequisites: None

Determines if the text has an automatic color set for it.

Usage

Text.**IsColorAutomatic**() As Boolean

Arguments

None

Return Values

As Boolean. True - there is an automatic color set for the text. False - no automatic color is set for the text.

For more information, see "SetAutomaticColor Method (Text Object)" on page 567.

SetAutomaticColor Method (Text Object)

Scope: Schematic editor

Object: Text Object

Prerequisites: None

Sets or unsets the color of a text object as the automatic color for text.

Usage

Text.**SetAutomaticColor**(byVal *bAutomatic* as Boolean)

Arguments

- bAutomatic

  The color that is set as the default for the object.

  If an object has automatic color, as determined by the IsColorAutomatic Method (Text Object), the actual color of the object is the default color for this type of object.

SetLocation Method (Text Object)

Scope: Schematic editor

Object: Text Object

Prerequisites: None

Specifies the coordinates of the text object.

Note:
All coordinates are measured in 100ths of an inch.

Usage

Text.**SetLocation**(ByVal *X* As Long, ByVal *Y* As Long)

Arguments

- X

  X coordinate of the text.
- Y

  Y coordinate of the text.

SetObjectColor Method (Text Object)

Scope: Schematic editor

Object: Text Object

Prerequisites: None

Sets the color in which the text is drawn.

Usage

Text.**SetObjectColor**(ByVal *newColor* as IColor)

Arguments

- Color

  The color assigned to the text. The new color is assigned as a Color object, as described in "CColor Object" on page 256.

Application Property (Text Object)

Scope: Schematic editor

Object: Text Object

Access: Read-Only

Prerequisites: None

Returns the Application Object.

Usage

Text.**Application**

Arguments

None

Return Values

IVdApp. The Application Object.

Description

See Application Object for more information.

Font Property (Text Object)

Scope: Schematic editor

Object: Text Object

Access: Read/Write

Prerequisites: None

Returns or sets the font in which the text is written.

Usage

Text.**Font** = VdFont

Arguments

None

Return Values

VdFont. The return/set type for this property. This is of the form VdFont Enum.

Orientation Property (Text Object)

Scope: Schematic editor

Object: Text Object

Access: Read/Write

Prerequisites: None

Returns or sets the orientation of the text.

Usage

Text.**Orientation** = VdOrientation

Arguments

None

Return Values

VdOrientation. The return/set type for this property. This is of the form VdOrientation Enum.

Origin Property (Text Object)

Scope: Schematic editor

Object: Text Object

Access: Read/Write

Prerequisites: None

Returns or sets the origin for the text.

Usage

Text.**Origin** = VdOrigin

Arguments

None

Return Values

VdOrigin. The return/set type for this property. This is of the form VdOrigin Enum.

Parent Property (Text Object)

Scope: Schematic editor

Object: Text Object

Access: Read-Only

Prerequisites: None

Returns the parent Block Object for the text.

Usage

Text.**Parent**

Arguments

None

Return Values

IVdBlock. The parent Block Object for the text.

Selected Property (Text Object)

Scope: Schematic editor

Object: Text Object

Access: Write-Only

Prerequisites: None

Sets the selection status for the text.

Usage

Text.**Selected** = True | False

Arguments

None

Return Values

True | False. The set type for this property. True - selects the text. False - deselects the text.

Size Property (Text Object)

Scope: Schematic editor

Object: Text Object

Access: Read/Write

Prerequisites: None

Returns or sets the size (height) of the text.

Usage

Text.**Size** = Long

Arguments

None

Return Values

Long. The value representing the size of the text.

TextString Property (Text Object)

Scope: Schematic editor

Object: Text Object

Access: Read/Write

Prerequisites: None

Returns or sets the text string; that is, the actual contents of the text.

Usage

Text.**TextString** = String

Arguments

None

Return Values

String. A string containing the actual contents of the text.

Type Property (Text Object)

Scope: Schematic editor

Object: Text Object

Access: Read-Only

Prerequisites: None

Returns VDTS_TEXT, indicating that the object is, in fact, text.

Usage

Text.**Type**

Arguments

None

Return Values

VdObjectType. The return type for this property. This is of the form VdObjectType Enum.

View Object

This object encapsulates the graphics display of a document (either a schematic or a symbol).

A View object may or may not be the active view (see "ActivateView Event (Application Object)" on page 120). The ActiveView is the last View to receive focus within the application.

The following table lists methods, properties and events of the View object with links to the respective reference pages:

Table 33. View Object Methods, Properties, and Events

Method, Property, or Event	Description
Activate Method (View Object)	Assigns the view from which this method is invoked as the active view.
AddAttributeMoveMode Method (View Object)	Adds an attribute to the currently active view attached to the cursor.
Application Method (View Object)	Returns the Application Object.
BufferCopy Method (View Object)	Copies the selected items to the buffer.
BufferCut Method (View Object)	Cuts the selected items to the buffer.
BufferPaste Method (View Object)	Pastes the selected items from the buffer.
BufferPasteXY Method (View Object)	Pastes the data from the buffer to the active view at the reference point specified. This method does not require user intervention.
ComputeMBB Method (View Object)	Sets the values of the four coordinate values to define the page bounding box.
Document Method (View Object)	Returns the ShematicSheetDocument Object contained by the view.
GetJointLocs Method (View Object)	Returns a string that contains the X,Y locations of the joints specified.
GetName Method (View Object)	Returns the hierarchical path to the block associated with this view.
GetSelectedNetName Method (View Object)	Returns the name of the selected net(s).
GetTopLevelDesignName Method (View Object)	Returns the block name associated with the top level hierarchical design for this view.
ModifyVisibility Method (View Object)	Changes the visibility of the target attributes where the attribute name string matches the string SelectString.
Query Method (View Object)	Returns a collection of objects based on type.
Refresh Method (View Object)	Causes the application to refresh the graphical depiction of the view. Call this method after graphical changes, moves, pastes, copies or deletions.
SelectbyName Method (View Object)	Selects an object by name.
SelectbyName2 Method (View Object)	Selects objects by name.

Table 33. View Object Methods, Properties, and Events (continued)

Method, Property, or Event	Description
SelectObject Method (View Object)	Selects an object by name.
SelectSegmentByJointLoc Method (View Object)	Selects a net segment based on the location and type of the lower joint in the segment.
SelectText Method (View Object)	Selects text strings that match the regular expression.
SetCenter Method (View Object)	Centers the view around the coordinates (maintaining the current zoom factor).
ViewFull Method (View Object)	Maximizes the view and redraws the graphical display.
ZoomIn Method (View Object)	Executes a zoom in.
ZoomOut Method (View Object)	Executes a zoom out.
ZoomSelect Method (View Object)	Centers the display on the selected item or items and adjusts the magnification and position of the image so that all the selected objects are in view. Xpedition Designer redraws the view after adjusting the viewport.
Block Property (View Object)	Returns a Block Object contained within the view.
TopBlock Property (View Object)	Returns the Block Object associated with the top level hierarchical design for this view.
Viewport Property (View Object)	Returns the graphics Viewport Object.
OnActivate Event (View Object)	Occurs when a view is being activated.
OnSelect Event (View Object)	Occurs when an object on a view is selected.

Activate Method (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Assigns the view from which this method is invoked as the active view.

Usage

View.**Activate**()

Arguments

None

AddAttributeMoveMode Method (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Adds an attribute to the currently active view attached to the cursor.

Usage

View.**AddAttributeMoveMode**(ByVal *AttributeString* As String, ByVal *Visibility* As VdVisibilityFlag)

Arguments

- AttributeString

  Attribute string (NAME=VALUE) to be added.

- Visibility

  Represents the visibility of the attribute once it has been added. This is of the form VdVisibilityFlag Enum.

Description

The attribute is attached to the cursor for the user to place. The attribute will move with the cursor until the user places it with a left mouse click.

Application Method (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Returns the Application Object.

Usage

View.**Application**() As IVdApp

Arguments

None

Return Values

As IVdApp. The Application Object.

Description

See Application Object for more information.

BufferCopy Method (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Copies the selected items to the buffer.

Usage

View.**BufferCopy**()

Arguments

None

Description

Copied items can be pasted with either the BufferPaste Method (View Object) or the BufferPasteXY Method (View Object). This is a good way to move data between schematics. It is not normally used to copy or move data on a single document.

BufferCut Method (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Cuts the selected items to the buffer.

Usage

View.**BufferCut**()

Arguments

None

Description

Cut items can later be pasted with either the BufferPaste Method (View Object) or the BufferPasteXY Method (View Object). This is a good way to move data between schematics.

BufferPaste Method (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Pastes the selected items from the buffer.

Note:
The programmer needs to notify the user that a paste operation is happening because does not.

Usage

View.**BufferPaste**()

Arguments

None

Description

BufferPaste starts the operation of pasting the data in the buffer to the active view. You must complete this operation by clicking on the location that is the reference point for the paste.

BufferPasteXY Method (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Pastes the data from the buffer to the active view at the reference point specified. This method does not require user intervention.

Usage

View.**BufferPasteXY**(ByVal *PasteX* As Long, ByVal *PasteY* As Long) As Boolean

Arguments

- PasteX

  X coordinate of the reference point for the paste operation.
- PasteY

  Y coordinate of the reference point for the paste operation.

Return Values

As Boolean. The return type for this method. True - the paste operation was successful. False - the paste operation could not be completed.

ComputeMBB Method (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Sets the values of the four coordinate values to define the page bounding box.

Usage

View.**ComputeMBB**(ByVal *OLEItems* As Boolean, *Xmin* As Long, *Ymin* As Long, *Xmax* As Long, *Ymax* As Long)

Arguments

- OLEItems

  Determines whether or not the bounding box is calculated to include any objects outside the page.

  True - the bounding box is calculated to include outside objects. False - the bounding box remains within the page.
- Xmin

  Specifies the minimum X value of the page bounding box.
- Ymin

  Specifies the minimum Y value of the page bounding box.
- Xmax

  Specifies the maximum X value of the page bounding box.
- Ymax

  Specifies the maximum Y value of the page bounding box.

Description

The page bounding box is normally the size of the page. However; if there is an object imbedded outside the normal page edges and OLEItems is TRUE the MBB will be adjusted to enclose this object as well.

Document Method (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Returns the ShematicSheetDocument Object contained by the view.

Usage

View.**Document**() As IVdSchematicSheetDocument

Arguments

None

Return Values

As IVdDoc. The SchematicSheetDocument Object.

Description

Automation objects are organized as trees and the root of the tree for the application is the IVdApp object. Navigation between objects in the tree involves getting or creating object pointers and then using them. Most objects support navigation to their parent object through the use of a parent pointer that they store when they are created. IVdView is different in that it provides a method for getting to the IVdDoc object that is its parent.

See SchematicSheetDocument Object for more information.

GetJointLocs Method (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Returns a string that contains the X,Y locations of the joints specified.

Note:
All coordinates are measured in 100ths of an inch.

Note:
Labels and attributes are assigned to segments, not to nets.

Usage

View.**GetJointLocs**(ByVal *AllOrSelected* As VdAllOrSelected, ByVal *JointType* As VdJointType) As String

Arguments

- AllOrSelected

  Nets are constructed from segments. A segment is normally uniquely specified by its end points. This argument specifies whether to consider the selected nets or only those specified. This is of the form VdAllOrSelected Enum.

- JointType

  Returns joints of the specifies types only. This is of the form VdJointType Enum.

Return Values

As String. A string containing the coordinates for the joints specified.

Description

This method returns a string containing the X,Y locations of all the joints specified. These values are in the order X Y X Y X Y ... and the values are space delimited. This method can be very useful for selecting a specific segment to attach a label or attribute.

GetName Method (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Returns the hierarchical path to the block associated with this view.

Usage

View.**GetName**(ByVal *Flag* As VdNameType) As String

Arguments

- Flag

  Specifies whether the full path to the block is returned, or just the short name. This is of the form VdNameType Enum.

Return Values

As String. A string containing the hierarchical path to the block associated with this view.

GetSelectedNetName Method (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Returns the name of the selected net(s).

Usage

View.**GetSelectedNetName**(ByVal *bRetFullPath* As Boolean, ByVal *bRetInternalName* As Boolean, [ByVal *Index* As Long = 1]) As String

Arguments

- bRetFullPath

  Specifies whether or not the full path of the net is returned: True - the hierarchical path to the net precedes the net name; False - only the net name is returned.

- bRetInternalName

  Specifies whether the internal net name is returned, or the user-assigned label: True - the internal net name is returned; False - the user-assigned label for the net is returned.

- Index

  (Optional). A number indicating one of the selected nets. The default value is 1.

Return Values

As String. A string containing the name of the selected net(s), or an empty string if no net is selected or if the index is out of range.

Examples

Find the names of multiple nets in the selected list.

```
Index = 1
For Each Net In ActiveView.Query(VDM_NET, VD_SELECTED)
    MsgBox ActiveView.GetSelectedNetName(FALSE, FALSE, Index)
    Index = Index + 1
Next
```

GetTopLevelDesignName Method (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Returns the block name associated with the top level hierarchical design for this view.

Note:
While the ActivateView Event (Application Object) is being executed, this method may return the incorrect design name because the context has not been set at that point. Consider using ActivateView2 Event (Application Object) instead to avoid this problem.

Usage

View.**GetTopLevelDesignName**() As String

Arguments

None

Return Values

As String. A string containing the block name associated with the top level hierarchical design for this view.

ModifyVisibility Method (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Changes the visibility of the target attributes where the attribute name string matches the string *SelectString*.

Usage

View.**ModifyVisibility**(ByVal *SelectString* As String, ByVal *Visibility* As VdVisibilityFlag, ByVal *ApplyToSelected* As VdAllOrSelected)

Arguments

- SelectString

  String that contains the target attribute name(s).

- Visibility

  Indicates the new visibility assigned. This is of the form VdVisibilityFlag Enum.

- ApplyToSelected

  When *ApplyToSelected* is TRUE, the search is restricted to selected attributes. Otherwise the search is applied to all unattached attributes in the view.

  This is of the form VdAllOrSelected Enum.

Query Method (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Returns a collection of objects based on type.

Usage

View.**Query**(ByVal *Flags* As VdObjectTypeMask, ByVal *Selected* As VdAllOrSelected) As IVdObjs

Arguments

- Flags

  Masks which may be combined to refine the query. This is of the form VdObjectTypeMask Enum.

- Selected

  Indicates whether the execution of this method applies to all objects, or only the selected objects. This is of the form VdAllOrSelected Enum.

Return Values

As IVdObjs. A collection of objects of a particular type.

Description

The Query method is very valuable for traversing all objects of a type. You can perform the query on all objects or on selected objects only.

Refresh Method (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Causes the application to refresh the graphical depiction of the view. Call this method after graphical changes, moves, pastes, copies or deletions.

Usage

View.**Refresh**()

Arguments

None

SelectbyName Method (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Selects an object by name.

Usage

View.**SelectByName**(ByVal *Name* As String) As Boolean

Arguments

- Name

  String to compare to the label on each object.

Return Values

As Boolean. The return type for this method. True - indicates that at least one object with a label matching *Name* was found. False - indicates that there were no objects found with a label matching *Name.*

Description

The method selects only those objects with labels that match the *Name* string and then places them on the selection list. The method flushes the existing selection list first.

SelectbyName2 Method (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Selects objects by name.

Usage

View.**SelectByName2**(ByVal *lpszObjName* As String, ByVal *bAddSelect* As Boolean) As Boolean

Arguments

- lpszObjName

  String to compare to the label of each object. To select an object, its label must match the *lpszObjName* string.

- bAddSelect

  Indicates whether or not to add the newly-selected items to the existing selection list.

  True - add the items to the existing selection list. False - creates a new selection list and adds the items to that selection list.

Return Values

As Boolean. True - indicates that at least one object with a label matching *lpszObjName* was found. False - indicates that there were no objects found with a label matching *lpszObjName*.

SelectObject Method (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Selects an object by name.

Usage

View.**SelectObject**(ByVal *ObjectType* As VdObjectType, ByVal *Expression* As String, ByVal *SelectOwner* As Boolean, ByVal *RegExp* As Boolean, ByVal *AddSelect* As Boolean) As Long

Arguments

- ObjectType

  Indicates which object types to consider. This is of the form "VdObjectType Enum" on page 733.

- Expression

  String expression that will be compared to the label or name on each object. This can be a regular expression.

- SelectOwner

  Specifies whether or not to select the parent of the target object.

  True - the parent object is selected. False - the parent object is not selected.

- RegExp

  Specifies whether or not to evaluate the *Expression* string or perform a simple string comparison.

  True - evaluate the *Expression* string normally. False - only performs a simple string comparison.

- AddSelect

  Indicates whether or not to add the newly-selected items to the existing selection list.

  True - the newly-selected items are added to the existing selection list. False - the newly-selected items are not added to the existing selection list; a new selection list is started.

Return Values

As Long. Returns zero (0). No objects are returned.

Description

This method provides a way to select objects, and optionally their owners, based on a regular expression. Use the method to restart the selection list or add to the existing list. Objects are selected by type in each search, but you can apply this function several times with different object types to do more general selections.

Examples

The example below selects all components with symbol name “cap_fxd_001_xfr.2”:

```
Dim res,itms

'select components with same symbol name
res=ActiveView.SelectObject(VDTS_COMPONENT, "cap_fxd_001_xfr.2", False,
   True, False)
MsgBox(res)’method returns 0

'get collection of selected components in the view
set itms=ActiveView.Query(VDM_COMP,VD_SELECTED)

if itms.count=0 then
    MsgBox("No Components Found")
else
  MsgBox(itms.count & " Components Found")
end if
```

SelectSegmentByJointLoc Method (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Selects a net segment based on the location and type of the lower joint in the segment.

Note:
Locations are defined as lower if they are closer to the location X=0, Y=0. It is possible for two segments to have the same lower joint location and the same joint type at that location. This is a rare condition usually resulting from pasting two copies on top of each other. In this case, only the first occurrence in Xpedition Designer's display list is selected.

Note:
All coordinates are measured in 100ths of an inch.

Usage

View.**SelectSegmentByJointLoc**(ByVal *XCoordinate* As Long, ByVal *YCoordinate* As Long, ByVal *JointType* As VdJointType) As Boolean

Arguments

- XCoordinate

  The X coordinate of the low joint of the segment.
- YCoordinate

  The Y coordinate of the low joint of the segment.
- JointType

  Defines the type of joint at this location to be considered. This is of the form VdJointType Enum.

Return Values

As Boolean. The return type for this method. True - the selection occurred. False - no selection occurred.

SelectText Method (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Selects text strings that match the regular expression.

Usage

View.**SelectText**(ByVal *Pattern* As String, ByVal *Type* As Long, ByVal *ApplyToSelected* As VdAllOrSelected)

Arguments

- Pattern

  Regular expression that will be compared to text strings.

- Type

  Indicates the kind of object to consider: VDTS_TEXT, VDTS_ATTRIBUTE, or VDTS_LABEL.

- ApplyToSelected

  Indicates whether to consider all objects or just the selected objects. This is of the form VdAllOrSelected Enum.

  If *ApplyToSelected* is False, then the comparison is to all the text in the view. If *ApplyToSelected* is True, then the comparison is restricted to the current selection list. Either way, the selection list will only contain the text objects that match the pattern after applying this method.

SetCenter Method (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Centers the view around the coordinates (maintaining the current zoom factor).

Usage

View.**SetCenter**(ByVal *X* As Long, ByVal *Y* As Long)

Arguments

- X

  X-coordinate of the view center.
- Y

  Y-coordinate of the view center.

ViewFull Method (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Maximizes the view and redraws the graphical display.

Usage

View.**ViewFull**()

Arguments

None

ZoomIn Method (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Executes a zoom in.

Usage

View.**ZoomIn**() As Boolean

Arguments

None

Return Values

As Boolean. The return type for this method. True - the zoom in operation was successful. False - the zoom in operation was not successful.

ZoomOut Method (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Executes a zoom out.

Usage

View.**ZoomOut**() As Boolean

Arguments

None

Return Values

As Boolean. The return type for this method. True - the zoom out operation was successful. False - the zoom out operation was not successful.

ZoomSelect Method (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Centers the display on the selected item or items and adjusts the magnification and position of the image so that all the selected objects are in view. Xpedition Designer redraws the view after adjusting the viewport.

Note:
If this method is called twice consecutively, Xpedition Designer may not always execute a redraw on the second call.

Usage

View.**ZoomSelect**() As Boolean

Arguments

None

Return Values

As Boolean. The return type for this method. True - the zoom operation was successful. False - the zoom operation was not successful.

Block Property (View Object)

Scope: Schematic editor

Object: View Object

Access: Read-Only

Prerequisites: None

Returns a Block Object contained within the view.

Usage

View.**Block**

Arguments

None

Return Values

IVdBlock. The Block Object.

Description

See Block Object for more information.

TopBlock Property (View Object)

Scope: Schematic editor

Object: View Object

Access: Read-Only

Prerequisites: None

Returns the Block Object associated with the top level hierarchical design for this view.

Note:
While the ActivateView Event (Application Object) is being executed, this property may return the incorrect block because the context has not been set at that point. Consider using ActivateView2 Event (Application Object) instead to avoid this problem.

Usage

View.**TopBlock**

Arguments

None

Return Values

IVdBlock. The Block Object associated with the top level hierarchical design for this view. Returns NULL if the view isn't a schematic, or if the top block isn't currently loaded in memory.

Viewport Property (View Object)

Scope: Schematic editor

Object: View Object

Access: Read-Only

Prerequisites: None

Returns the graphics Viewport Object.

Usage

View.**Viewport**

Arguments

None

Return Values

IViewport. The Viewport Object.

OnActivate Event (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Occurs when a view is being activated.

Usage

Sub View_OnActivate()

Arguments

None

OnSelect Event (View Object)

Scope: Schematic editor

Object: View Object

Prerequisites: None

Occurs when an object on a view is selected.

Usage

Sub View_OnSelect(ByVal *View* As IVdView)

Arguments

- View

  The View Object that contains the selected object.

Viewport Object

The Viewport object encapsulates the Xpedition Designer graphics interface. Using the methods and properties associated with this object allows you to draw graphics onto a View.

These graphics are temporary and are not stored in the iCDB database.

The following table lists methods and properties of the Viewport object with links to the respective reference page.

Table 34. Viewport Methods and Properties

Method or Property	Description
Arc Method (Viewport Object)	Draws an arc, specified by three points.
Arrow Method (Viewport Object)	Draws an arrow from one point to another.
Box Method (Viewport Object)	Draws a box.
Circle Method (Viewport Object)	Draws a circle at the specified location with the specified radius.
Ellipse Method (Viewport Object)	Draws an ellipse centered at X,Y with radii XRadius, YRadius.
EraseRectangle Method (Viewport Object)	Erases a rectangle.
GetObjectColor Method (Viewport Object)	Gets the color in which the viewport is drawn.
Line Method (Viewport Object)	Draws a line from one point to another.
PixelRectangle Method (Viewport Object)	Returns the viewport rectangle in terms of pixel coordinates.
PixelToUser Method (Viewport Object)	Translates pixel coordinates into user coordinates.
Point Method (Viewport Object)	Draws a pixel at a specified location.
PolyLine Method (Viewport Object)	Draws a polygon line.
SetClipRectangle Method (Viewport Object)	Sets the clipping region for drawing in the form of a Rect Object.
SetObjectColor Method (Viewport Object)	Sets the color in which the viewport is drawn.
Spline Method (Viewport Object)	Draws a spline with specified control points.
Text Method (Viewport Object)	Draws a text string at the specified location.
UserRectangle Method (Viewport Object)	Returns a viewport rectangle using the specified coordinates.
UserToPixel Method (Viewport Object)	Translates user coordinates into pixel coordinates.
FillStyle Property (Viewport Object)	Returns or sets the fill style for polygons, circles and boxes.
LineCap Property (Viewport Object)	Returns or sets the style of the end point of a line.

Table 34. Viewport Methods and Properties (continued)

Method or Property	Description
LineJoin Property (Viewport Object)	Returns or sets the line join type for polygons.
LinePattern Property (Viewport Object)	Returns or sets the line pattern type.
LineThickness Property (Viewport Object)	Returns or sets line thickness.
RasterMode Property (Viewport Object)	Returns or sets the raster mode.
TextAngle Property (Viewport Object)	Sets the text angle.
TextFont Property (Viewport Object)	Sets the font used for text.
TextSize Property (Viewport Object)	Sets the size (height) of the text.

Arc Method (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Prerequisites: None

Draws an arc, specified by three points.

Usage

Viewport.**Arc**(ByVal *X1* As Long, ByVal *Y1* As Long, ByVal *X2* As Long, ByVal *Y2* As Long, ByVal *X3* As Long, ByVal *Y3* As Long)

Arguments

- X1

  X coordinate of the first point.
- Y1

  Y coordinate of the first point.
- X2

  X coordinate of the second point.
- Y2

  Y coordinate of the second point.
- X3

  X coordinate of the third point.
- Y3

  Y coordinate of the third point.

Arrow Method (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Prerequisites: None

Draws an arrow from one point to another.

Usage

Viewport.**Arrow**(ByVal *X1* As Long, ByVal *Y1* As Long, ByVal *X2* As Long, ByVal *Y2* As Long, ByVal *Arrowhead* As VdArrowType)

Arguments

- X1

  X coordinate of the first point.
- Y1

  Y coordinate of the first point.
- X2

  X coordinate of the second point.
- Y2

  Y coordinate of the second point.
- Arrowhead

  Specifies the type of arrowhead used. This is of the form VdArrowType Enum.

Box Method (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Prerequisites: None

Draws a box.

Usage

Viewport.**Box**(ByVal *Left* As Long, ByVal *Top* As Long, ByVal *Right* As Long, ByVal *Bottom* As Long)

Arguments

- Left

  Leftmost X coordinate.
- Top

  Topmost Y coordinate.
- Right

  Rightmost X coordinate.
- Bottom

  Bottommost Y coordinate.

Circle Method (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Prerequisites: None

Draws a circle at the specified location with the specified radius.

Usage

Viewport.**Circle**(ByVal *X* As Long, ByVal *Y* As Long, ByVal *Radius* As Long)

Arguments

Note:
All coordinates are measured in 100ths of an inch.

- X

  X coordinate of the center of the circle.
- Y

  Y coordinate of the center of the circle.
- Radius

  The length of the radius of the circle.

Ellipse Method (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Prerequisites: None

Draws an ellipse centered at X,Y with radii XRadius, YRadius.

Usage

Viewport.**Ellipse**(ByVal *X* As Long, ByVal *Y* As Long, ByVal *XRadius* As Long, ByVal *YRadius* As Long)

Arguments

- X

  X coordinate of the ellipse center.
- Y

  Y coordinate of the ellipse center.
- XRadius

  Radius of the ellipse along the X axis.
- YRadius

  Radius of the ellipse along the Y axis.

EraseRectangle Method (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Prerequisites: None

Erases a rectangle.

Usage

Viewport.**EraseRectangle**(ByVal *Left* As Long, ByVal *Top* As Long, ByVal *Right* As Long, ByVal *Bottom* As Long)

Arguments

- Left

  Leftmost X coordinate of the rectangle to be erased.
- Top

  Topmost Y coordinate of the rectangle to be erased.
- Right

  Rightmost X coordinate of the rectangle to be erased.
- Bottom

  Bottommost Y coordinate of the rectangle to be erased.

GetObjectColor Method (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Prerequisites: None

Gets the color in which the viewport is drawn.

Usage

Viewport.**GetObjectColor**() As IColor

Arguments

None

Return Values

As Color. See "CColor Object" on page 256.

Line Method (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Prerequisites: None

Draws a line from one point to another.

Usage

Viewport.**Line**(ByVal *X1* As Long, ByVal *Y1* As Long, ByVal *X2* As Long, ByVal *Y2* As Long)

Arguments

- X1

  X coordinate of the first point.
- Y1

  Y coordinate of the first point.
- X2

  X coordinate of the second point.
- Y2

  Y coordinate of the second point.

PixelRectangle Method (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Prerequisites: None

Returns the viewport rectangle in terms of pixel coordinates.

Usage

Viewport.**PixelRectangle**(*Left* As Variant, *Top* As Variant, *Right* As Variant, *Bottom* As Variant)

Arguments

- Left

  Leftmost X coordinate of the rectangle.
- Top

  Topmost Y coordinate of the rectangle.
- Right

  Rightmost X coordinate of the rectangle.
- Bottom

  Bottommost Y coordinate of the rectangle.

PixelToUser Method (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Prerequisites: None

Translates pixel coordinates into user coordinates.

Usage

Viewport.**PixelToUser**(*XCoordinate* As Variant, *YCoordinate* As Variant)

Arguments

- XCoordinate

  X coordinate value.
- YCoordinate

  Y coordinate value.

Point Method (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Prerequisites: None

Draws a pixel at a specified location.

Usage

Viewport.**Point**(ByVal *X* As Long, ByVal *Y* As Long)

Arguments

Note:
All coordinates are measured in 100ths of an inch.

- X

  X coordinate for the pixel.
- Y

  Y coordinate for the pixel.

PolyLine Method (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Prerequisites: None

Draws a polygon line.

Usage

Viewport.**PolyLine**(ByVal *ArrayOfXValues* As Variant, ByVal *ArrayOfYValues* As Variant) As Boolean

Arguments

- ArrayOfXValues

  A single dimension array of X ordinate values.
- ArrayOfYValues

  A single dimension array of Y ordinate values.

Return Values

As Boolean. True - the polygon line was successfully drawn. False - the polygon line could not be drawn.

Examples

Draw a polygon line.

```
Dim XArray(10)
Dim YArray(10)

For I=0 To 10-1
     XArray(I) = ...
  YArray(I) = ...
Next I

ActiveView.Viewport.PolyLine XArray, YArray
```

SetClipRectangle Method (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Prerequisites: None

Sets the clipping region for drawing in the form of a Rect Object.

Usage

Viewport.**SetClipRectangle**(ByVal *Left* As Long, ByVal *Top* As Long, ByVal *Right* As Long, ByVal *Bottom* As Long) As IVdRect

Arguments

- Left

  Leftmost X coordinate.
- Top

  Topmost Y coordinate.
- Right

  Rightmost X coordinate.
- Bottom

  Bottommost Y coordinate.

Return Values

As IVdRect. The Rect Object.

SetObjectColor Method (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Prerequisites: None

Sets the color in which the viewport is drawn.

Usage

Viewport.**SetObjectColor**(*newColor* As IColor)

Arguments

- newColor

  The color assigned to the viewport. The new color is assigned as a Color object, as described in "CColor Object" on page 256.

Spline Method (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Prerequisites: None

Draws a spline with specified control points.

Usage

Viewport.**Spline**(ByVal *Type* As VdSplineType, ByVal *Order* As VdSplineOrder, ByVal *Granularity* As Integer, ByVal *XArrayOfPoints* As Variant, ByVal *YArrayOfPoints* As Variant, ByVal *Arrowhead* As VdArrowType, ByVal *ArrowWidth* As Integer, ByVal *ArrowLength* As Integer, ByVal *ArrowFill* As VdFillStyle)

Arguments

- Type

  Specify the spline type. This argument uses the form VdSplineType Enum.

- Order

  Specifies the order of the spline. Order controls the continuity of the curve. A value of SHARP causes the curve to come close to but may not actually pass through a control point. If you actually want the curve to pass through the control point you can “emphasize” any control point by simply replicating the coordinates of that point in the input control point list. A value of SMOOTH makes the curve more continuous and a value of ROUND makes it even more continuous. This argument uses the form VdSplineOrder Enum.

- Granularity

  Granularity controls the smoothness of the curve. The spline is computed as a series of individual points from which line segments are drawn. Granularity controls exactly how many line segments are actually drawn between successive control points. A low value produces a low-resolution spline which may look almost look like a polygon line and higher numbers produce smoother curves. Since this number is multiplied by the number of control points try to keep it as low as possible. A value of 10 is a good place to start.

- XArrayOfPoints

  Array of X ordinate values. Must be a single dimensioned array.

- YArrayOfPoints

  Array of Y ordinate values. Must be a single dimensioned array.

- Arrowhead

  Indicates how arrow heads should be drawn. This argument uses the form VdArrowType Enum.

- ArrowWidth

  Specifies the arrowhead width.

- ArrowLength

  Specifies the arrowhead length.

- ArrowFill

Specifies the fill style for the arrowhead.

Text Method (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Prerequisites: None

Draws a text string at the specified location.

Usage

Viewport.**Text**(ByVal *X* As Long, ByVal *Y* As Long, ByVal *String* As String, ByVal *Flags* As VdTextFlags)

Arguments

Note:
All coordinates are measured in 100ths of an inch.

- X

  X coordinate.
- Y

  Y coordinate.
- String

  The actual text message that is to be drawn.
- Flags

  Text flags, of the form VdTextFlags Enum.

UserRectangle Method (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Prerequisites: None

Returns a viewport rectangle using the specified coordinates.

Usage

Viewport.**UserRectangle**(*Left* As Variant, *Top* As Variant, *Right* As Variant, *Bottom* As Variant)

Arguments

- Left

  Leftmost X coordinate of the rectangle.
- Top

  Topmost Y coordinate of the rectangle.
- Right

  Rightmost X coordinate of the rectangle.
- Bottom

  Bottommost Y coordinate of the rectangle.

UserToPixel Method (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Prerequisites: None

Translates user coordinates into pixel coordinates.

Usage

Viewport.**UserToPixel**(*XCoordinate* As Variant, *YCoordinate* As Variant)

Arguments

- XCoordinate

  X coordinate.
- YCoordinate

  Y coordinate.

FillStyle Property (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Access: Read/Write

Prerequisites: None

Returns or sets the fill style for polygons, circles and boxes.

Usage

Viewport.**FillStyle** = VdFillStyle

Arguments

None

Return Values

VdFillStyle. The return/set type for this property. This is of the form VdFillStyle Enum.

LineCap Property (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Access: Read/Write

Prerequisites: None

Returns or sets the style of the end point of a line.

Usage

Viewport.**LineCap** = VdLineCap

Arguments

None

Return Values

VdLineCap. The return/set type for this property. This is of the form VdLineCap Enum.

LineJoin Property (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Access: Read/Write

Prerequisites: None

Returns or sets the line join type for polygons.

Usage

Viewport.**LineJoin** = VdLineJoin

Arguments

None

Return Values

VdLineJoin. The return/set type for this property. This is of the form VdLineJoin Enum.

LinePattern Property (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Access: Read/Write

Prerequisites: None

Returns or sets the line pattern type.

Usage

Viewport.**LinePattern** = VdLinePattern

Arguments

None

Return Values

VdLinePattern. The return/set type for this property. This is of the form VdLinePattern Enum.

LineThickness Property (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Access: Read/Write

Prerequisites: None

Returns or sets line thickness.

Usage

Viewport.**LineThickness** = Long

Arguments

None

Return Values

Long. A value that represents the thickness of the line.

RasterMode Property (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Access: Read/Write

Prerequisites: None

Returns or sets the raster mode.

Usage

Viewport.**RasterMode** = VdRasterop

Arguments

None

Return Values

VdRasterop. The return/set type for this property. This is of the form VdRasterop Enum.

TextAngle Property (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Access: Write-Only

Prerequisites: None

Sets the text angle.

Usage

Viewport.**TextAngle** = Long

Arguments

None

Return Values

Long. A value that represents the angle of the text.

TextFont Property (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Access: Write-Only

Prerequisites: None

Sets the font used for text.

Usage

Viewport.**TextFont** = VdFont

Arguments

None

Return Values

VdFont. The set type for this property. This is of the form VdFont Enum.

TextSize Property (Viewport Object)

Scope: Schematic editor

Object: Viewport Object

Access: Write-Only

Prerequisites: None

Sets the size (height) of the text.

Usage

Viewport.**TextSize** = Long

Arguments

None

Return Values

Long. A value that represents the size of the text.

Chapter 4
Schematic Editor Object Collections

The following table includes summary information for each collection you can access in Xpedition Designer automation.

Table 35. Xpedition Designer Collections

Collection	Description
DesignerDataManagementAdditionalProperties Collection	You can access EDM managed designs using the DesignerDataManagementAdditionalProperties collection. The following table lists methods of the DesignerDataManagementAdditionalProperties collection with links to the respective reference pages.
DesignerDataManagementEntities Collection	You can access EDM managed designs using the DesignerDataManagementEntities collection. The following table lists methods and properties of the DesignerDataManagementEntities collection with links to the respective reference pages.
HDLSourceDocuments Collection	This object represents a collection of HDLSourceDocument Objects.
SchematicSheetDocuments Collection	Documents is an automation collection that represents the set of Xpedition Designer schematic documents.
StringCollection Collection	This object represents a collection of strings that exist on a Xpedition Designer schematic. This collection provides methods for identifying and/or removing strings from the schematic, but not otherwise modifying them.
StringList Collection	This object is a collection of strings that exist on a Xpedition Designer schematic. This collection has a well-defined interface that allows for removing/adding/iterating any of the elements that compose the collection.

DesignerDataManagementAdditionalPropert ies Collection

You can access EDM managed designs using the DesignerDataManagementAdditionalProperties collection. The following table lists methods of the DesignerDataManagementAdditionalProperties collection with links to the respective reference pages.

This is a collection of "DesignerDataManagementAdditionalProperty" on page 351 objects.

Table 36. DesignerDataManagementAdditionalProperties Collection Methods

Method or Property	Description
Count Method (DesignerDataManagementAdditionalProperties Collection)	Returns the number of additional property objects in this collection.
Item Method (DesignerDataManagementAdditionalProperties Collection)	Returns the additional property of a given index in the collection.
Remove Method (DesignerDataManagementAdditionalProperties Collection)	Removes an object from the collection.

Count Method (DesignerDataManagementAdditionalProperties Collection)

Scope: Schematic editor; Embedded Client (EDM)

Collection: DesignerDataManagementAdditionalProperties Collection

Prerequisites: None

Returns the number of additional property objects in this collection.

Usage

DesignerDataManagementAdditionalProperties.**Count**() As Long

Arguments

None

Return Values

Long. The number of objects the collection contains.

Examples

This example outputs the total number of data management additional property objects.

```
MsgBox DesignerDataManagementAdditionalProperties.Count()
```

Item Method (DesignerDataManagementAdditionalProperties Collection)

Scope: Schematic editor; Embedded Client (EDM)

Collection: DesignerDataManagementAdditionalProperties Collection

Prerequisites: None

Returns the additional property of a given index in the collection.

Usage

DesignerDataManagementAdditionalProperties.**Item** (ByVal *vIndex* As Variant) As IMGCDesignerDataManagementAdditionalProperty

Arguments

- vIndex

  A zero-based index of the additional properties.

Return Values

"IMGCDesignerDataManagementAdditionalProperty" on page 351. The additional property of the given index in the collection.

Remove Method (DesignerDataManagementAdditionalProperties Collection)

Scope: Schematic editor; Embedded Client (EDM)

Collection: DesignerDataManagementAdditionalProperties Collection

Prerequisites: None

Removes an object from the collection.

Usage

DesignerDataManagementAdditionalProperties.**Remove**(ByVal vIndex As Variant) As Nothing

Arguments

- vIndex

  The index of the "DesignerDataManagementAdditionalProperty" on page 351 to remove from the collection.

Return Values

Nothing

DesignerDataManagementEntities Collection

You can access EDM managed designs using the DesignerDataManagementEntities collection. The following table lists methods and properties of the DesignerDataManagementEntities collection with links to the respective reference pages.

Table 37. DesignerDataManagementEntities Collection Methods and Properties

Method or Property	Description
Count Method (DesignerDataManagementEntities Collection)	Returns the number of objects in this collection.
Item Method (DesignerDataManagementEntities Collection)	Returns a DesignerDataManagementEntity object contained in the collection.
Remove Method (DesignerDataManagementEntities Collection)	Removes an object from the collection.

Count Method (DesignerDataManagementEntities Collection)

Scope: Schematic editor; Embedded Client (EDM)

Collection: DesignerDataManagementEntities Collection

Prerequisites: None

Returns the number of objects in this collection.

Usage

DesignerDataManagementEntities.**Count**() As Long

Arguments

None

Return Values

Long. A value representing the number of objects in the collection.

Item Method (DesignerDataManagementEntities Collection)

Scope: Schematic editor; Embedded Client (EDM)

Collection: DesignerDataManagementEntities Collection

Prerequisites: None

Returns a DesignerDataManagementEntity object contained in the collection.

Usage

DesignerDataManagementEntities.**Item**(ByVal vIndex As Variant) As IMGCDesignerDataManagementEntity

Arguments

- vIndex

  Numerical index of the DesignerDataManagementEntity object to retrieve from the collection.

Return Values

IMGCDesignerDataManagementEntity. The DesignerDataManagementEntity Object.

Remove Method (DesignerDataManagementEntities Collection)

Scope: Schematic editor; Embedded Client (EDM)

Collection: DesignerDataManagementEntities Collection

Prerequisites: None

Removes an object from the collection.

Usage

DesignerDataManagementEntities.**Remove**(ByVal vIndex As Variant) As Nothing

Arguments

- vIndex

  The index of the DesignerDataManagementEntity Object to remove from the collection.

Return Values

Nothing

HDLSourceDocuments Collection

This object represents a collection of HDLSourceDocument Objects.

The following table lists methods and properties of the HDLSourceDocuments collection with links to the respective reference pages. For more information, see HDLSourceDocument Object.

Table 38. HDLSourceDocuments Collection Methods and Properties

Method or Property	Description
Item Method (HDLSourceDocuments Collection)	Returns an HDLSourceDocument Object contained in the collection.
New Method (HDLSourceDocuments Collection)	Creates a new HDLSourceDocument Object based on the specified type, then adds it to this collection.
Open Method (HDLSourceDocuments Collection)	Opens an HDLSource Document Object derived from the document path and adds it to this collection.
Remove Method (HDLSourceDocuments Collection)	Removes a source file from the collection, using the index.
RemoveAll Method (HDLSourceDocuments Collection)	Removes all HDLSourceDocument Objects from the collection.
SaveAll Method (HDLSourceDocuments Collection)	Saves all HDL source documents in the HDLSourceDocuments Collection.
Count Property (HDLSourceDocuments Collection)	Returns the number of HDLSourceDocument Objects contained in the collection.

Item Method (HDLSourceDocuments Collection)

Scope: Xpedition Designer schematic editor

Collection: HDLSourceDocuments Collection

Access: Read-Only

Prerequisites: None

Returns an HDLSourceDocument Object contained in the collection.

Usage

HDLSourceDocuments.**Item**(ByVal *Index* As Variant) As IHDLSourceDocument

Arguments

- *Index*

  Numerical index of the HDLSourceDocument Object to retrieve from the collection.

Note:
The document count in an HDLSourceDocuments collection begins at 1, not 0. This is primarily used by the C++ interface since it does not support a For Each type of iterator.

Return Values

IHDLSourceDocument. The HDLSourceDocument Object.

Examples

Using C++:

```
For Index=1 To HDLSourceDocuments.Count
   Set Doc = HDLSourceDocuments.Item(Index)
Next
```

Using VBScript:

```
For Each Doc In HDLSourceDocuments
   DocName = Doc.Name
Next
```

New Method (HDLSourceDocuments Collection)

Scope: Xpedition Designer schematic editor

Collection: HDLSourceDocuments Collection

Prerequisites: None

Creates a new HDLSourceDocument Object based on the specified type, then adds it to this collection.

Usage

HDLSourceDocuments.**New**(ByVal *DocType* As VdSourceDocumentType) As IHDLSourceDocument

Arguments

- *DocType*

  0 - plain text document

  1 - VHDL source document

  2 - Verilog source document

  3 - SPICE source document

Return Values

As IHDLSourceDocument. The HDLSourceDocument Object.

Open Method (HDLSourceDocuments Collection)

Scope: Xpedition Designer schematic editor

Collection: HDLSourceDocuments Collection

Prerequisites: None

Opens an HDLSource Document Object derived from the document path and adds it to this collection.

Usage

HDLSourceDocuments.**Open**(ByVal *DocumentPath* As String) As IHDLSourceDocument

Arguments

- DocumentPath

  The path to the source document to open.

Return Values

As IHDLSourceDocument. The HDLSourceDocument Object.

Remove Method (HDLSourceDocuments Collection)

Scope: Schematic editor

Collection: HDLSourceDocuments Collection

Prerequisites: None

Removes a source file from the collection, using the index.

Usage

HDLSourceDocuments.**Remove**(ByVal *Index* As Long)

Arguments

- *Index*

  A value representing the index number of theHDLSourceDocument Object to remove from the collection.

RemoveAll Method (HDLSourceDocuments Collection)

Scope: Schematic editor

Collection: HDLSourceDocuments Collection

Prerequisites: None

Removes all HDLSourceDocument Objects from the collection.

Usage

HDLSourceDocuments.**RemoveAll**()

Arguments

None

Description

See HDLSourceDocument Object for more information.

SaveAll Method (HDLSourceDocuments Collection)

Scope: Xpedition Designer schematic editor

Collection: HDLSourceDocuments Collection

Prerequisites: None

Saves all HDL source documents in the HDLSourceDocuments Collection.

Usage

HDLSourceDocuments.**SaveAll**()

Arguments

None

Count Property (HDLSourceDocuments Collection)

Scope: Xpedition Designer schematic editor

Collection: HDLSourceDocuments Collection

Access: Read-Only

Prerequisites: None

Returns the number of HDLSourceDocument Objects contained in the collection.

Usage

HDLSourceDocuments.**Count** = Long

Arguments

None

Return Values

Long. A value representing the number of HDLSourceDocument Objects the collection contains.

SchematicSheetDocuments Collection

Documents is an automation collection that represents the set of Xpedition Designer schematic documents.

The following table lists methods and properties of the SchematicSheetDocuments collection with links to the respective reference pages. For more information, see SchematicSheetDocument Object.

Table 39. SchematicSheetDocuments Collection Methods and Properties

Method or Property	Description
Close Method (SchematicSheetDocuments Collection)	Closes all the currently open documents.
CopyToClipboard Method (SchematicSheetDocuments Collection)	Copies sheets from a schematic to the clipboard.
DeleteSheet Method (SchematicSheetDocuments Collection)	Deletes a specified sheet from a specified schematic.
GetAvailableSchematics Method (SchematicSheetDocuments Collection)	Returns the collection of all schematic document names for the current project.
GetAvailableSheets Method (SchematicSheetDocuments Collection)	Returns the collection of sheet names for a given schematic document.
InsertSheet Method (SchematicSheetDocuments Collection)	Inserts a specified sheet into a specified schematic.
IsSymbolUnderEdit Method (SchematicSheetDocuments Collection)	Checks whether the symbol has local changes that have not been updated in the design.
Item Method (SchematicSheetDocuments Collection)	Returns a SchematicSheetDocument object contained in the collection.
Open Method (SchematicSheetDocuments Collection)	Opens an existing schematic document and adds it to the collection.
Open_Hierarchically Method (SchematicSheetDocuments Collection)	Opens an existing schematic document hierarchically and adds it to the collection.
OpenSymbol Method (SchematicSheetDocuments Collection)	Opens the symbol of a given name to edit.
PasteFromClipboard Method (SchematicSheetDocuments Collection)	Pastes the contents of the clipboard into the specified schematic, possibly replacing sheets within the schematic.
Application Property (SchematicSheetDocuments Collection)	Returns the Application Object.

Table 39. SchematicSheetDocuments Collection Methods and Properties (continued)

Method or Property	Description
Count Property (SchematicSheetDocuments Collection)	Returns the number of objects contained in the collection.
Parent Property (SchematicSheetDocuments Collection)	Returns or the parent collection of the collection.

Close Method (SchematicSheetDocuments Collection)

Scope: Xpedition Designer schematic editor

Collection: SchematicSheetDocuments Collection

Prerequisites: None

Closes all the currently open documents.

Note:
Be sure you have saved all document changes prior to this call and have released all Automation references to these documents or their sub-objects.

Usage

SchematicSheetDocuments.**Close**()

Arguments

None

CopyToClipboard Method (SchematicSheetDocuments Collection)

Scope: Xpedition Designer schematic editor

Collection: SchematicSheetDocuments Collection

Prerequisites: None

Copies sheets from a schematic to the clipboard.

Usage

SchematicSheetDocuments.**CopyToClipoard**(ByVal *SchematicName* As String, ByVal *SheetsToCopy* As IStringList, ByVal *srcPath* As IStringList)

Arguments

- *SchematicName*

  The name of the schematic document containing the sheets to be copied to the clipboard.
- *SheetsToCopy*

  A list of one or more sheets within the specified schematic to be copied to the clipboard.
- *srcPath*

  The hierarchical path to the schematic document.

Examples

This example copies two schematic sheets to the clipboard, then pastes them into the “nic” symbol.

Figure 4. CopyToClipboard

Note:
Mentor Graphics recommends that you use COM versioning syntax in script examples that use GetObject and CreateObject. Without COM versioning, the script will access the last installation to which the release switcher pointed.

```
Set Doc = app.SchematicSheetDocuments.Open("Schematic1", "1")

'sheets to copy Schematic1/1/11/(1,2)
set sheets = createobject("viewdraw.stringlist")
sheets.Append("1")
sheets.Append("2")

Set pathFrom = createobject("viewdraw.stringlist")
pathFrom.Append("Schematic1")
pathFrom.Append("1")
pathFrom.Append("11")

app.SchematicSheetDocuments.CopyToClipboard "11", sheets, pathFrom

msgbox "copied"

' destination Schematic1/2/nic
Set pathTo = createobject("viewdraw.stringlist")
pathTo.Append("Schematic1")
pathTo.Append("2")
pathTo.Append("nic")

set sheets2bereplaced = createobject("viewdraw.stringlist")
```

```
app.SchematicSheetDocuments.PasteFromClipboard "nic", pathTo,
 sheets2bereplaced
```

Figure 5. PasteFromClipboard

DeleteSheet Method (SchematicSheetDocuments Collection)

Scope: Xpedition Designer schematic editor

Collection: SchematicSheetDocuments Collection

Prerequisites: None

Deletes a specified sheet from a specified schematic.

Usage

SchematicSheetDocuments.**DeleteSheet**(ByVal *SchematicName* As String, ByVal *SheetNumber* As Long) As Boolean

Arguments

- *SchematicName*

  The name of the schematic from which to delete the sheet.

- *SheetNumber*

  The number of the sheet within the specified schematic that is to be deleted.

Return Values

Boolean. True - the schematic sheet was deleted (success). False - the schematic sheet could not be deleted (failure).

GetAvailableSchematics Method (SchematicSheetDocuments Collection)

Scope: Xpedition Designer schematic editor

Collection: SchematicSheetDocuments Collection

Prerequisites: None

Returns the collection of all schematic document names for the current project.

Usage

SchematicSheetDocuments.**GetAvailableSchematics**() As IStringList

Arguments

None

Return Values

As IStringList. The collection of SchematicSheetDocument Object names.

GetAvailableSheets Method (SchematicSheetDocuments Collection)

Scope: Xpedition Designer schematic editor

Collection: SchematicSheetDocuments Collection

Prerequisites: None

Returns the collection of sheet names for a given schematic document.

Usage

SchematicSheetDocuments.**GetAvailableSheets**(ByVal *Schematic* As String) As IStringList

Arguments

- *Schematic*

  The name of the schematic document for which to retrieve sheet names.

Return Values

As StringList. The collection of sheet names.

InsertSheet Method (SchematicSheetDocuments Collection)

Scope: Xpedition Designer schematic editor

Collection: SchematicSheetDocuments Collection

Prerequisites: None

Inserts a specified sheet into a specified schematic.

Usage

SchematicSheetDocuments.**InsertSheet**(ByVal *SchematicName* As String, ByVal *SheetNumber* As String) As Boolean

Arguments

- *SchematicName*

  A string containing the name of the schematic into which the sheet is inserted.
- *SheetNumber*

  A string containing the number that is assigned to the inserted sheet.

Return Values

As Boolean. True - The schematic sheet was successfully inserted. False - The schematic sheet could not be inserted.

IsSymbolUnderEdit Method (SchematicSheetDocuments Collection)

Scope: Schematic editor - Embedded symbol editor (ESE)

Collection: SchematicSheetDocuments Collection

Prerequisites: None

Checks whether the symbol has local changes that have not been updated in the design.

Usage

SchematicSheetDocuments.**IsSymbolUnderEdit** (ByVal *sSymbolName* As String, ByVal *sSymbolExtension* As String) As Integer

Arguments

- sSymbolName

  String. A string that contains the symbol name.
- sSymbolExtension

  String. A string that contains the symbol extension.

Return Values

Integer. Returns a value of 1 (true) if the symbol has uncommitted changes; otherwise, returns a value of 0 (false).

Examples

```
Set app = CreateObject("Viewdraw.Application")
Scripting.AddTypeLibrary("Viewdraw.Application")
app.OpenProject("PROJECT_NAME.prj")

Set test = app.SchematicSheetDocuments.Open("SCHEMATIC_NAME", 1)
Set myView = app.ActiveView
Set all_components = myView.query(VDM_COMP, VD_ALL)

For Each component_instance In all_components
   Set my_symbol = component_instance.SymbolBlock
   edited = app.SchematicSheetDocuments.IsSymbolUnderEdit( _
      my_symbol.getname(0), "1")
   If edited = true Then
      msgbox my_symbol.getname(0) + " has uncommitted changes"
   End If
Next
```

Item Method (SchematicSheetDocuments Collection)

Scope: Xpedition Designer schematic editor

Collection: SchematicSheetDocuments Collection

Prerequisites: None

Returns a SchematicSheetDocument object contained in the collection.

Usage

SchematicSheetDocuments.**Item**(ByVal *Index* As Long) As Document

Arguments

- Index

  Numerical index of the Document object to retrieve from the collection.

Return Values

Document. The SchematicSheetDocument Object.

Open Method (SchematicSheetDocuments Collection)

Scope: Xpedition Designer schematic editor

Collection: SchematicSheetDocuments Collection

Prerequisites: None

Opens an existing schematic document and adds it to the collection.

Usage

SchematicSheetDocuments.**Open**(ByVal *SchematicName* As String, ByVal *SheetName* As String) As Document

Arguments

- *SchematicName*

  A string containing the name of the schematic document.

- *SheetName*

  A string containing the name of the schematic sheet.When this argument references a non-existing sheet, the Open method creates a new sheet.

Return Values

As Document. The SchematicSheetDocument Object.

Open_Hierarchically Method (SchematicSheetDocuments Collection)

Scope: Xpedition Designer schematic editor

Collection: SchematicSheetDocuments Collection

Prerequisites: None

Opens an existing schematic document hierarchically and adds it to the collection.

Usage

SchematicSheetDocuments.**Open_Hierarchically**(ByVal *SchematicName* As String, ByVal *SheetName* As String, ByVal *HierPath* As IStringList) As Document

Arguments

- *SchematicName*

  A string containing the name of the schematic document.
- *SheetName*

  A string containing the name of the schematic sheet.
- *HierPath*

  A string containing the hierarchical path to the specified schematic document.

Return Values

As Document. The SchematicSheetDocument Object.

OpenSymbol Method (SchematicSheetDocuments Collection)

Scope: Schematic editor - Embedded symbol editor (ESE)

Collection: SchematicSheetDocuments Collection

Prerequisites: None

Opens the symbol of a given name to edit.

Usage

SchematicSheetDocuments.**OpenSymbol** (*sSymbolName* As String, *sSymbolExtension* As String) As IVdSchematicSheetDocument

Arguments

- sSymbolName

  String. A string that contains the symbol name.
- sSymbolExtension

  String. A string that contains the symbol extension.

Return Values

IVdSchematicSheetDocument. A document that represents one sheet of a schematic.

Examples

```
Set app = CreateObject("Viewdraw.Application")
Scripting.AddTypeLibrary("Viewdraw.Application")
app.OpenProject("PROJECT_NAME.prj")
Set ese_test = app.SchematicSheetDocuments.OpenSymbol("my_loc_sym", "1")
```

PasteFromClipboard Method (SchematicSheetDocuments Collection)

Scope: Xpedition Designer schematic editor

Collection: SchematicSheetDocuments Collection

Prerequisites: None

Pastes the contents of the clipboard into the specified schematic, possibly replacing sheets within the schematic.

Usage

SchematicSheetDocuments.**PasteFromClipoard**(ByVal *SchematicName* As String, ByVal *dstPath* As IStringList, ByVal *SheetsToBeReplaced* As IStringList)

Arguments

- *SchematicName*

  The name of the schematic document into which the clipboard contents are pasted.

- *dstPath*

  The hierarchical path to the schematic document.

- *SheetsToReplace*

  A list of one or more sheets within the specified schematic that are replaced by the contents of the clipboard.

Examples

Please see the example under the topic CopyToClipboard Method (SchematicSheetDocuments Collection).

Application Property (SchematicSheetDocuments Collection)

Scope: Xpedition Designer schematic editor

Collection: SchematicSheetDocuments Collection

Access: Read-Only

Prerequisites: None

Returns the Application Object.

Usage

SchematicSheetDocuments.**Application**

Arguments

None

Return Values

IVdApp. The Application Object.

Count Property (SchematicSheetDocuments Collection)

Scope: Xpedition Designer schematic editor

Collection: SchematicSheetDocuments Collection

Access: Read-Only

Prerequisites: None

Returns the number of objects contained in the collection.

Usage

SchematicSheetDocuments.**Count**

Arguments

None

Return Values

Long. The number of SchematicSheetDocument Objects the collection contains.

Parent Property (SchematicSheetDocuments Collection)

Scope: Xpedition Designer schematic editor

Collection: SchematicSheetDocuments Collection

Access: Read-Only

Prerequisites: None

Returns or the parent collection of the collection.

Usage

SchematicSheetDocuments.**Parent**

Arguments

None

Return Values

IVdApp. The return type for this property.

StringCollection Collection

This object represents a collection of strings that exist on a Xpedition Designer schematic. This collection provides methods for identifying and/or removing strings from the schematic, but not otherwise modifying them.

This collection is to be used only as a return object in scripts.

The following table lists methods and properties of the StringCollection collection with links to the respective reference pages.

Table 40. StringCollection Collection Methods and Properties

Method or Property	Description
Item Method (StringCollection Collection)	Returns an element (specified by its index number) from the collection.
Remove Method (StringCollection Collection)	Removes an element (specified by its index number) from the collection.
Count Property (StringCollection Collection)	Returns the number of elements in the collection.

Item Method (StringCollection Collection)

Scope: Xpedition Designer schematic editor

Collection: StringCollection Collection

Prerequisites: None

Returns an element (specified by its index number) from the collection.

Usage

StringCollection.**Item**(ByVal *Index* As Long) As String

Arguments

- *Index*

  The index number of the element in the collection. Indexing starts at 1.

Return Values

As String. A string that is part of the collection.

Remove Method (StringCollection Collection)

Scope: Xpedition Designer schematic editor

Collection: StringCollection Collection

Prerequisites: None

Removes an element (specified by its index number) from the collection.

Usage

StringCollection.**Remove**(ByVal *Index* As Long)

Arguments

- *Index*

  The index number of the element to remove from the collection. Index numbering starts at 1.

Count Property (StringCollection Collection)

Scope: Xpedition Designer schematic editor

Collection: StringCollection Collection

Access: Read-Only

Prerequisites: None

Returns the number of elements in the collection.

Usage

StringCollection.**Count**

Arguments

None

Return Values

Long. A value representing the number of strings in the collection.

StringList Collection

This object is a collection of strings that exist on a Xpedition Designer schematic. This collection has a well-defined interface that allows for removing/adding/iterating any of the elements that compose the collection.

This collection can be used as either a parameter of as a return object in scripts. It is constrained by specific semantics only in the context of a given method.

For example, GetAvailableSheets Method (SchematicSheetDocuments Collection) uses the StringList collection to return the names of available sheets; in this case the collection is of strings that denote that names of schematic sheets.

The following table lists methods of the StringList collection with links to the respective reference pages.

Table 41. StringList Collection Methods

Method	Description
Append Method (StringList Collection)	Adds an element to the end of the collection.
Clear Method (StringList Collection)	Removes all elements from the collection.
GetCount Method (StringList Collection)	Returns the number of elements in the collection.
GetItem Method (StringList Collection)	Returns a string (specified by Index) that is contained in the collection.
Insert Method (StringList Collection)	Inserts an element into the collection at the position specified by the index argument.
Remove Method (StringList Collection)	Removes an element specified by its index position from the collection.

Append Method (StringList Collection)

Scope: Xpedition Designer schematic editor

Collection: StringList Collection

Prerequisites: None

Adds an element to the end of the collection.

Usage

StringList.**Append**(ByVal *sItem* As String)

Arguments

- *sItem*

  The element to add.

Clear Method (StringList Collection)

Scope: Xpedition Designer schematic editor

Collection: StringList Collection

Prerequisites: None

Removes all elements from the collection.

Usage

StringList.**Clear**()

Arguments

None

GetCount Method (StringList Collection)

Scope: Xpedition Designer schematic editor

Collection: StringList Collection

Prerequisites: None

Returns the number of elements in the collection.

Usage

StringList.**GetCount**() As Long

Arguments

None

Return Values

Long. A value representing the number of string lists in the collection.

GetItem Method (StringList Collection)

Scope: Xpedition Designer schematic editor

Collection: StringList Collection

Prerequisites: None

Returns a string (specified by *Index*) that is contained in the collection.

Usage

StringList.**GetItem**(ByVal *Index* As Long) As String

Arguments

- *Index*

  Long value specifying the index of the element in the collection. Indexing starts at 1.

Return Values

String. A string representing an element in the collection as specified by the *Index* argument.

Insert Method (StringList Collection)

Scope: Xpedition Designer schematic editor

Collection: StringList Collection

Prerequisites: None

Inserts an element into the collection at the position specified by the index argument.

Usage

StringList.**Insert**(ByVal *sItem* As String, ByVal *Index* As Long)

Arguments

- *sItem*

  The string object that is to be added to the collection.

- *Index*

  Index of the element in the collection. Indexing starts at 1.

Remove Method (StringList Collection)

Scope: Xpedition Designer schematic editor

Collection: StringList Collection

Prerequisites: None

Removes an element specified by its index position from the collection.

Usage

StringList.**Remove**(ByVal *Index* As Long)

Arguments

- *Index*

  Index of the element in the collection. Indexing starts at 1.

Chapter 5
Schematic Editor Enumerated Types

The Xpedition Designer schematic editor supports a wide range of enumerated types.

Schematic Editor Enumerated Types Summary

Schematic Editor Enumerated Types Summary

The table below includes summary information for each enumerated type you can access with Xpedition Designer Automation.

Table 42. Xpedition Designer Enumerated Types

Enumerated Type	Description
DesignerErrCode Enum	Designer error code constants.
PinMappingType Enum	Specifies a pin mapping type.
PropertyMappingType Enum	Specifies a property mapping type.
ScopeReplaceSymbol Enum	Specifies the replace symbol scope.
ShellEventID Enum	Specifies the shell event ID.
ToolType Enum	Specifies a tool type.
VdAllOrSelected Enum	Specifies whether to consider all items or only those that are currently selected.
VdAnnoObject Enum	Specifies the type of object to be annotated.
VdAnnoPos Enum	Specifies an origin point for an annotation.
VdAppEventDispatchID Enum	Specifies the condition that causes an event to occur.
VdArcPoint Enum	Specifies a point that defines an arc.
VdArrowType Enum	Specifies the type of arrowheads used for drawing a line.
VdBoolean Enum	Specifies the boolean value of an argument or condition.
VdBusOrWire Enum	Specifies whether a net is a bus or a wire.
VdCompInstanceForwardPCB Enum	Specifies whether the component should be forward annotated to the PCB.
VdCorner Enum	Specifies a corner point for an object.
VdCreateTime Enum	Specifies the time at which a notification appears when an object it placed.
VdDataType Enum	Specifies the block data type.
VdDMDumpPropertiesFilter Enum	Data management entity dump properties filter constants.
VdDMEntityAccessType Enum	Specifies the data management entity access type.
VdDMEntityType Enum	Specifies the data management entity type.
VdDocumentAccess Enum	Specifies the document access types.
VdFillStyle Enum	Specifies a fill style.
VdFont Enum	Specifies a font style.
VdGradientType Enum	Specifies a gradient type.
VdJointType Enum	Specifies a joint type.
VdLabelVisibility Enum	Specifies visibility status.
VdLineCap Enum	Specifies the line cap used for lines.
VdLineJoin Enum	Specifies the style for a line join.
VdLinePattern Enum	Specifies the pattern used for lines.
VdLineStyle Enum	Specifies the style used for lines.
VdNameType Enum	Specifies the format that is displayed for net names.

Table 42. Xpedition Designer Enumerated Types (continued)

Enumerated Type	Description
VdNotifyFlag Enum	Specifies the events that trigger flags.
VdObjectClass Enum	Specifies an object class.
VdObjectType Enum	Specifies an object type.
VdObjectTypeMask Enum	Specifies an object type mask (usually for selection purposes).
VdOnOff Enum	Specifies mode settings.
VdOpenMode Enum	Specifies the mode for an opened block.
VdOrientation Enum	Specifies an orientation style.
VdOrigin Enum	Specifies a point of origin.
VdParamMode Enum	Specifies various modes for the Xpedition Designer session.
VdParamValue Enum	Specifies various values used in the Xpedition Designer session.
VdPEFlowMode Enum	Specifies the PE flow mode.
VdPinEndType Enum	Specifies a pin end type.
VdRasterop Enum	Specifies raster operations.
VdScope Enum	Specifies the scope.
VdSegmentEndType Enum	Specifies the segment end type.
VdSelectionType Enum	Specifies the selection notification types.
VdSense Enum	Specifies the sense for labels and pins.
VdSheetSize Enum	Specifies sheet size.
VdSide Enum	Specifies the pin side.
VdSilentMode Enum	Specifies silent mode severity settings.
VdSourceDocumentType Enum	Specifies source document types.
VdSplineOrder Enum	Specifies spline order.
VdSplineType Enum	Specifies spline type.
VdSymbolType Enum	Specifies symbol block type.
VdTextFlags Enum	Specifies text flags.
VdUpdateOOScope Enum	Sets the scope of an update.
VdUpdateOtherObjects Enum	Determines which objects are updated.
VdVisibilityFlag Enum	Specifies attribute visibility flags.
VdWhichJoint Enum	Specifies joint end points.
WindowStyles Enum	Specifies the command window style.

DesignerErrCode Enum

Scope: Schematic editor

Prerequisites: None.

Designer error code constants.

Usage

DesignerErrCode.*Constant*

Arguments

- **DesignerErrCodeDataManagementAuthorizationFailed**

  The data management authorization failed. The numerical value for this constant is -2147220987 (&H80040205).

- **DesignerErrCodeDataManagementChangeServerWhileEditing**

  Change server while editing. The numerical value for this constant is -2147220980 (&H8004020C).

- **DesignerErrCodeDataManagementEntityTypeProductMismatch**

  A data management entity type product mismatch. The numerical value for this constant is -2147220982 (&H8004020A).

- **DesignerErrCodeDataManagementInvalidDirectory**

  An invalid directory. The numerical value for this constant is -2147220985 (&H80040207).

- **DesignerErrCodeDataManagementNoWritePermission**

  No write permission. The numerical value for this constant is -2147220984 (&H80040208).

- **DesignerErrCodeDataManagementOtherError**

  An other, unspecified error. The numerical value for this constant is -2147220983 (&H80040209).

- **DesignerErrCodeDataManagementReadonlySetting**

  The setting is read-only. The numerical value for this constant is -2147220977 (&H8004020F).

- **DesignerErrCodeDataManagementServiceFailed**

  The data management service failed. The numerical value for this constant is -2147220986 (&H80040206).

- **DesignerErrCodeDataManagementServiceNotConnected**

  The data management service is not connected. The numerical value for this constant is -2147220988 (&H80040204).

- **DesignerErrCodeDataManagementUnknownSetting**

  The setting is unknown. The numerical value for this constant is -2147220979 (&H8004020D).

- **DesignerErrCodeDataManagementValueNotInRange**

  The value is out of range. The numerical value for this constant is -2147220978 (&H8004020E).

- **DesignerErrCodeFunctionNotSupported**

  The function is not supported, The numerical value for this constant is -2147220981 (&H8004020B).

- **DesignerErrCodeInternalError**

  An internal error. The numerical value for this constant is -2147220990 (&H80040202).

- **DesignerErrCodeInvalidParameter**

  An invalid parameter. The numerical value for this constant is -2147220991 (&H80040201).

- **DesignerErrOutOfRange**

  Out of range error. The numerical value for this constant is -2147220989 (&H80040203).

PinMappingType Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None

Specifies a pin mapping type.

Usage

PinMappingType.*Constant*

Arguments

- **PinMT_ByName**

  Map pins by name. The numerical value for this constant is 1.
- **PinMT_ByPinNumber**

  Map pins by pin number. The numerical value for this constant is 2.

PropertyMappingType Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None

Specifies a property mapping type.

Usage

PropertyMappingType.*Constant*

Arguments

- PropMT_LibraryOnly

  The numerical value for this constant is 1.
- PropMT_LibraryWins

  The numerical value for this constant is 3.
- PropMT_SchematicWins

  The numerical value for this constant is 2.

ScopeReplaceSymbol Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None

Specifies the replace symbol scope.

Usage

ScopeReplaceSymbol.*Constant*

Arguments

- **SRS_Board**

  The numerical value for this constant is 2.
- **SRS_Project**

  The numerical value for this constant is 1.
- **SRS_Schematic**

  The numerical value for this constant is 3.
- **SRS_Selection**

  The numerical value for this constant is 5.
- **SRS_Sheet**

  The numerical value for this constant is 4.

ShellEventID Enum

Scope: Schematic editor

Prerequisites: None

Specifies the shell event ID.

Usage

ShellEventID.*Constant*

Arguments

- **ShellEventIDExit**

  The numerical value for this constant is 3.
- **ShellEventIDStart**

  The numerical value for this constant is 4.
- **ShellEventIDStdError**

  The numerical value for this constant is 2.
- **ShellEventIDStdOutput**

  The numerical value for this constant is 1.

ToolType Enum

Scope: Schematic editor

Prerequisites: None

Specifies a tool type.

Usage

ToolType.*Constant*

Arguments

- **ShellExecuteValue**

  The numerical value for this constant is 1.

VdAllOrSelected Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies whether to consider all items or only those that are currently selected.

Usage

VdAllOrSelected.*Constant*

Arguments

- VD_ALL

  The numerical value for this constant is 0.
- VD_SELECTED

  The numerical value for this constant is 1.

VdAnnoObject Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies the type of object to be annotated.

Usage

VdAnnoObject.*Constant*

Arguments

- AnnoObjCOMPONENT

  Annotate a component. The numerical value for this constant is 0.
- AnnoObjCOMPONENT_PIN

  Annotate a component pin. The numerical value for this constant is 1.
- AnnoObjNET

  Annotate a net. The numerical value for this constant is 2.
- AnnoObjPIN

  Annotate a pin. The numerical value for this constant is 3.
- AnnoObjWINDOW

  Annotate a window. The numerical value for this constant is 4.

VdAnnoPos Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies an origin point for an annotation.

Usage

VdAnnoPos.*Constant*

Arguments

- AnnoPosCENTER

  Annotate at the center. The numerical value for this constant is 4.
- AnnoPosLOWERLEFT

  Annotate to the lower left. The numerical value for this constant is 0.
- AnnoPosLOWERRIGHT

  Annotate to the lower right. The numerical value for this constant is 1.
- AnnoPosUPPERLEFT

  Annotate to the upper left. The numerical value for this constant is 2.
- AnnoPosUPPERRIGHT

  Annotate to the upper right. The numerical value for this constant is 3.

VdAppEventDispatchID Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies the condition that causes an event to occur.

Usage

VdAppEventDispatchID.*Constant*

Arguments

- AppEventACTIVATE_VIEW

  The event occurs when a view is activated. The numerical value for this constant is 3. See "ActivateView Event (Application Object)" on page 120.

- AppEventACTIVATE_VIEW2

  The event occurs when a view (that is passed as an argument) is activated. The numerical value for this constant is 20 (&H14). See "ActivateView2 Event (Application Object)" on page 121.

- AppEventAFTER_DOCUMENT_OPENED

  The event occurs when a document has been opened and the graphical display drawn. The numerical value for this constant is 6. See "AfterDocumentOpened Event (Application Object)" on page 122.

- AppEventAFTER_PRINT_PROJECT

  The event occurs after the completion of a print project operation. The numerical value for this constant is 28 (&H1C). See "AfterPrintProject Event (Application Object)" on page 123.

- AppEventAFTER_SHEET_READ

  The event occurs after a new sheet is read into memory. The numerical value for this constant is 19 (&H13). See "AfterSheetRead Event (Application Object)" on page 124.

- AppEventAFTER_SHEET_REREAD

  The event occurs after a new sheet is reread into memory. The numerical value for this constant is 37 (&H25). See "AfterSheetReRead Event (Application Object)" on page 125.

- AppEventAFTER_SYMBOL_DEFINITION_REFRESH

  The event occurs after a symbol definition has been refreshed. The numerical value for this constant is 55 (&H37).

- AppEventAFTERDOCUMENTSAVED

  The event occurs after a document is saved. The numerical value for this constant is 18 (&H12).

- AppEventBEFORE_DOCUMENT_CLOSED

  The event occurs just before a document is closed. The numerical value for this constant is 54 (&H36). See "AfterDocumentOpened Event (Application Object)" on page 122.

- AppEventBEFORE_DOCUMENT_OPENED

The event occurs when a document is opened, but before the graphical display is drawn. The numerical value for this constant is 5. See “AfterDocumentOpened Event (Application Object)” on page 122.

- AppEventBEFORE_PRINT_PROJECT

  The event occurs just before a print project operation is initiated. The numerical value for this constant is 27 (&H1B). See “BeforePrintProject Event (Application Object)” on page 127.

- AppEventBEFOREDOCUMENTSAVED

  The event occurs just before a document is saved. The numerical value for this constant is 17 (&H11).

- AppEventBEFORE_PROJECT_CHANGED

  The event occurs just before a project is changed. The numerical value for this constant is 41 (&H29). See “BeforeProjectChanged Event (Application Object)” on page 128.

- AppEventBLOCK_MODIFIED

  The event occurs just before a block is modified. The numerical value for this constant is 22 (&H16). See “BlockModified Event (Application Object)” on page 130.

- AppEventCLOSE

  The event occurs just before a project is closed. The numerical value for this constant is 14 (&HE).

- AppEventCREATE_OBJECT

  The event occurs whenever any object is created. The numerical value for this constant is 9. See “CreateObject Event (Application Object)” on page 131.

- AppEventDBCONFIG_MODIFIED

  The event occurs whenever the database configuration has been modified. The numerical value for this constant is 56 (&H38).

- AppEventDEACTIVATE_VIEW

  The event occurs whenever a view is deactivated (whenever the window loses focus). The numerical value for this constant is 4. See “DeactivateView Event (Application Object)” on page 132.

- AppEventDEACTIVATE_VIEW2

  The event occurs whenever a view (which is passes as an argument) is deactivated (whenever the window loses focus). The numerical value for this constant is 21 (&H15). See “DeactivateView2 Event (Application Object)” on page 133.

- AppEventDELETE

  The event occurs whenever an object is deleted. The numerical value for this constant is 12 (&HC). See “Delete Event (Application Object)” on page 134.

- AppEventDOCUMENT_CLOSE

  The event occurs whenever a document is closed. The numerical value for this constant is 33 (&H21). See “DocumentClose Event (Application Object)” on page 135.

- AppEventDOCUMENT_MODIFIED

  The event occurs whenever a document has been modified. The numerical value for this constant is 8.

- AppEventLOCKREQUEST

The event occurs when there is a lock request. The numerical value for this constant is 15. See "LockRequest Event (Application Object)" on page 136.

- AppEventMOUSEMOVED

  The event occurs when the mouse is moved in a View. The numerical value for this constant is 13. See "MouseMoved Event (Application Object)" on page 137.

- AppEventPAINT_REGION

  The event occurs when a View is painted. The numerical value for this constant is 26 (&H1A). See "PaintRegion Event (Application Object)" on page 138.

- AppEventPRINT_FILE

  The event occurs after a project is changed. The numerical value for this constant is 29 (&H1D). See "PrintFile Event (Application Object)" on page 139.

- AppEventPRODUCTION_LIBRARY_LIMITATIONS_CHANGED

  The event occurs after the user modifies the production library limitation setting. The numerical value for this constant is 57 (&H39). See "ProductionLibraryLimitationsChanged Event (Application Object)" on page 140.

- AppEventPROJECT_CHANGED

  The event occurs just before a print project operation is initiated. The numerical value for this constant is 40 (&H28). See "ProjectChanged Event (Application Object)" on page 141.

- AppEventSELECT

  The event occurs when an object is selected or unselected. The numerical value for this constant is 10. See "Select Event (Application Object)" on page 143.

- AppEventSHUTDOWN

  The event occurs when Xpedition Designer shuts down. The numerical value for this constant is 2. See "Shutdown Event (Application Object)" on page 144.

- AppEventSOURCEDOCUMENT_SAVE

  The event occurs when a source document is saved. The numerical value for this constant is 36 (&H24). See "SourceDocumentSave Event (Application Object)" on page 145.

- AppEventSOURCE_FILE_MODIFIED

  The event occurs when an existing source document is modified and saved. The numerical value for this constant is 39 (&H27). See "SourceFileModified Event (Application Object)" on page 146.

- AppEventSTARTUP

  The event occurs when Xpedition Designer starts. The numerical value for this constant is 1. See "Startup Event (Application Object)" on page 147.

- AppEventSYMOL_PREVIEWED

  The event occurs when the symbol previewer displays a new symbol. The numerical value for this constant is 35 (&H23). See "SymbolPreviewed Event (Application Object)" on page 148.

- AppEventUNLOCK

  The event occurs when a block is being unlocked. The numerical value for this constant is 16 (&H10). See "Unlock Event (Application Object)" on page 149.

VdArcPoint Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies a point that defines an arc.

Usage

VdArcPoint.*Constant*

Arguments

- VDARC_PT_ONE

  The first point that defines an arc. The numerical value for this constant is 0.
- VDARC_PT_THREE

  The third point that defines an arc. The numerical value for this constant is 2.
- VDARC_PT_TWO

  The second point that defines an arc. The numerical value for this constant is 1.

VdArrowType Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies the type of arrowheads used for drawing a line.

Usage

VdArrowType.*Constant*

Arguments

- VGARROWNONE

  No arrowhead is drawn. The numerical value for this constant is 3.
- VGARROWPOINT1

  An arrowhead is drawn at the first point defining the line. The numerical value for this constant is 0.
- VGARROWPOINT12

  Arrowheads are drawn at both points that define the line. The numerical value for this constant is 2.
- VGARROWPOINT2

  An arrowhead is drawn at the second point defining the line. The numerical value for this constant is 1.

VdBoolean Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies the boolean value of an argument or condition.

Usage

VdBoolean.*Constant*

Arguments

- VD_FALSE

  The argument or condition is false. The numerical value for this constant is 0.
- VD_TRUE

  The argument or condition is true. The numerical value for this constant is 1.

VdBusOrWire Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies whether a net is a bus or a wire.

Usage

VdBusOrWire.*Constant*

Arguments

- VD_BUS

  The net is a bus. The numerical value for this constant is 1.
- VD_WIRE

  The net is a wire. The numerical value for this constant is 0.

VdCompInstanceForwardPCB Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies whether the component should be forward annotated to the PCB.

Usage

VdCompInstanceForwardPCB.*Constant*

Arguments

- VdCompInstance_FALSE

  The component should not be forward annotated.

  The numerical value for this constant is 1.
- VdCompInstance_FromSymbol

  Value on the symbol specifies whether the component should be forward annotated.

  The numerical value for this constant is 2.
- VdCompInstance_TRUE

  The component should be forward annotated.

  The numerical value for this constant is 0.

VdCorner Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies a corner point for an object.

Usage

VdCorner.*Constant*

Arguments

- VDLOWERLEFT

  The lower left corner for the object. The numerical value for this constant is 0.
- VDUPPERRIGHT

  The upper right corner for an object. The numerical value for this constant is 1.

VdCreateTime Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies the time at which a notification appears when an object it placed.

Usage

VdCreateTime.*Constant*

Arguments

- VDCREATE_AFTER_NOTIFY

  The notification appears after the object is placed. The numerical value for this constant is 1.
- VDCREATE_BEFORE_NOTIFY

  The notification is appears before the object is placed. The numerical value for this constant is 0.

VdDataType Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies the block data type.

Usage

VdDataType.*Constant*

Arguments

- VDDT_DOCUMENTATION

  The block data type is a document. The numerical value for this constant is 2.
- VDDT_PIN

  The block data type is a pin. The numerical value for this constant is 4.
- VDDT_SCHEMATIC

  The block data type is schematic. The numerical value for this constant is 0.
- VDDT_SYMBOL

  The block data type is symbol. The numerical value for this constant is 1.
- VDDT_WIRELIST

  The block data type is wire list. The numerical value for this constant is 3.

VdDMDumpPropertiesFilter Enum

Scope: Schematic editor; Embedded Client (EDM)

Prerequisites: None.

Data management entity dump properties filter constants.

Usage

VdDMDumpPropertiesFilter.*Constant*

Arguments

- VDDMDUMPFILTER_ALL

  All metadata assigned to this object. The numerical value for this constant is 0.
- VDDMDUMPFILTER_CUSTOM

  All custom metadata created by this user. The numerical value for this constant is 2.
- VDDMDUMPFILTER_DEFINED_AS_ARGUMENT

  All metadata that have parameter Argument set to Yes. The numerical value for this constant is 3.
- VDDMDUMPFILTER_VISIBLE

  All visible metadata. The numerical value for this constant is 1.

VdDMEntityAccessType Enum

Scope: Schematic editor; Embedded Client (EDM)

Prerequisites: None.

Specifies the data management entity access type.

Usage

VdDMEntityAccessType.*Constant*

Arguments

- VDACCES_NONE

  The access type is None. The numerical value for this constant is 0.
- VDACCES_READONLY

  The access type is Read-Only. The numerical value for this constant is 1.
- VDACCES_READWRITE

  The access type is Read/Write. The numerical value for this constant is 2.

VdDMEntityType Enum

Scope: Schematic editor; Embedded Client (EDM)

Prerequisites: None.

Specifies the data management entity type.

Usage

VdDMEntityType.*Constant*

Arguments

- VDES_LAYOUT

  The entity type is Layout. The numerical value for this constant is 1.
- VDES_SCHEMATIC

  The entity type is Schematic. The numerical value for this constant is 2.
- VPADSPRO_DES_SCHEMATIC

  The entity type is PADS Pro Design Schematic. The numerical value for this constant is 256 (&H100)
- VSYS_DESIGN

  The entity type is Design. The numerical value for this constant is 4.

VdDocumentAccess Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies the document access types.

Usage

VdDocumentAccess.*Constant*

Arguments

- VDDA_BLOCK_WRITABLE

  The numerical value for this constant is 0.
- VDDA_OAT_READ_ONLY

  The numerical value for this constant is 6.
- VDDA_OAT_WRITABLE

  The numerical value for this constant is 2.
- VDDA_SCH_READ_LOCK

  Schematic file is locked. The numerical value for this constant is 9.
- VDDA_SCH_READ_ONLY

  The schematic file is read-only. The numerical value for this constant is 5.
- VDDA_SCH_WRITABLE

  The schematic file is writable. The numerical value for this constant is 1.
- VDDA_SYM_READ_LOCK

  The numerical value for this constant is 10.
- VDDA_SYM_READ_ONLY

  The numerical value for this constant is 7.
- VDDA_SYM_WRITABLE

  The numerical value for this constant is 3.
- VDDA_WIR_READ_ONLY

  The numerical value for this constant is 8.
- VDDA_WIR_WRITABLE

  The numerical value for this constant is 4.

VdFillStyle Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies a fill style.

Usage

VdFillStyle.*Constant*

Arguments

- VDFILL_DIAGDN1

  The fill style is diagonal, down, style 1. The numerical value for this constant is 2.
- VDFILL_DIAGDN2

  The fill style is diagonal, down, style 2. The numerical value for this constant is 5.
- VDFILL_DIAGUP1

  The fill style is diagonal, up, style 1. The numerical value for this constant is 6.
- VDFILL_DIAGUP2

  The fill style is diagonal, up, style 2. The numerical value for this constant is 3.
- VDFILL_GREY04

  The fill style is gray, 4%. The numerical value for this constant is 15 (&HF).
- VDFILL_GREY08

  The fill style is gray, 8%. The numerical value for this constant is 4.
- VDFILL_GREY50

  The fill style is gray, 50%. The numerical value for this constant is 13 (&HD).
- VDFILL_GREY92

  The fill style is gray, 92%. The numerical value for this constant is 14 (&HE).
- VDFILL_GRID1

  The fill style is grid style 1. The numerical value for this constant is 10 (&HA).
- VDFILL_GRID2

  The fill style is grid style 2. The numerical value for this constant is 9.
- VDFILL_HOLLOW

  The fill style is hollow. The numerical value for this constant is 0.
- VDFILL_HORIZ

  The fill style is horizontal. The numerical value for this constant is 7.
- VDFILL_SOLID

  The fill style is solid. The numerical value for this constant is 1.

- VDFILL_VERT

  The fill style is vertical. The numerical value for this constant is 8.
- VDFILL_X1

  The fill style is X, style 1. The numerical value for this constant is 12 (&HC).
- VDFILL_X2

  The fill style is X, style 2. The numerical value for this constant is 11 (&HB).

VdFont Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies a font style.

Usage

VdFont.*Constant*

Arguments

- VDFONT_FIXED

  A fixed font. The numerical value for this constant is 0.
- VDFONT_GOTHIC

  Gothic font. The numerical value for this constant is 9.
- VDFONT_KANJI

  Kanji Do not document. The numerical value for this constant is 11 (&HB).
- VDFONT_OLD_ENGLISH

  Old English font. The numerical value for this constant is 10 (&HA).
- VDFONT_PLOT

  Plotter font. The numerical value for this constant is 12 (&HC).
- VDFONT_ROMAN

  Roman font. The numerical value for this constant is 1.
- VDFONT_ROMAN_B

  Roman bold font. The numerical value for this constant is 3.
- VDFONT_ROMAN_BI

  Roman bold italic font. The numerical value for this constant is 4.
- VDFONT_ROMAN_I

  Roman italic font. The numerical value for this constant is 2.
- VDFONT_SANS_SERIF

  Sans serif font. The numerical value for this constant is 5.
- VDFONT_SANS_SERIF_B

  Sans serif bold font. The numerical value for this constant is 7.
- VDFONT_SCRIPT

  Script font. The numerical value for this constant is 6.
- VDFONT_SCRIPT_B

  Bold script font. The numerical value for this constant is 8.

VdGradientType Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies a gradient type.

Usage

VdGradientType.*Constant*

Arguments

- VDGRADIENT_DIAGONAL1

  A “diagonal 1” gradient. The numerical value for this constant is 3.
- VDGRADIENT_DIAGONAL2

  A “diagonal 2” gradient. The numerical value for this constant is 4.
- VDGRADIENT_HORIZONTAL

  A horizontal gradient. The numerical value for this constant is 2.
- VDGRADIENT_NONE

  No gradient. The numerical value for this constant is 0.
- VDGRADIENT_VERTICAL

  A vertical gradient. The numerical value for this constant is 1.

VdJointType Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies a joint type.

Usage

VdJointType.*Constant*

Arguments

- VDJT_BUS

  A bus joint. The numerical value for this constant is 6.
- VDJT_BUSCORNER

  A bus joint which has a 90 degree bend. The numerical value for this constant is 9.
- VDJT_BUSPIN

  A bus joint which connects to a pin. The numerical value for this constant is 8.
- VDJT_BUSSINGLE

  A dangling bus joint. The numerical value for this constant is 7.
- VDJT_BUSSOLDER

  A bus solder dot. The numerical value for this constant is 11 (&HB).
- VDJT_BUSSTRAIGHT

  A bus joint which is between two straight segments. The numerical value for this constant is 10 (&HA).
- VDJT_CORNER

  A net joint which has a 90 degree bend. The numerical value for this constant is 3.
- VDJT_LONER

  A net joint. The numerical value for this constant is 0.
- VDJT_PIN

  A net joint that connects to a pin. The numerical value for this constant is 2.
- VDJT_SINGLE

  A dangling net joint. The numerical value for this constant is 1.
- VDJT_SOLDER

  A net solder dot. The numerical value for this constant is 5.
- VDJT_STRAIGHT

  A net joint which is between two straight segments. The numerical value for this constant is 4.

VdLabelVisibility Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies visibility status.

Usage

VdLabelVisibility.*Constant*

Arguments

- VDLABELINVISIBLE

  Invisible. The numerical value for this constant is 0.
- VDLABELVISIBLE

  Visible. The numerical value for this constant is 1.

VdLineCap Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies the line cap used for lines.

Usage

VdLineCap.*Constant*

Arguments

- VGLINECAPBUTT

  The numerical value for this constant is 0.
- VGLINECAPNOT_LAST

  The numerical value for this constant is 3.
- VGLINECAPPROJECTING

  The numerical value for this constant is 2.
- VGLINECAPROUND

  The numerical value for this constant is 1.

VdLineJoin Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies the style for a line join.

Usage

VdLineJoin.*Constant*

Arguments

- VGLINEJOINBEVEL

  The numerical value for this constant is 2.
- VGLINEJOINMITER

  The numerical value for this constant is 0.
- VGLINEJOINROUND

  The numerical value for this constant is 1.

VdLinePattern Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies the pattern used for lines.

Usage

VdLinePattern.*Constant*

Arguments

- VGLINEPATTERNBIGDASH

  The numerical value for this constant is 4.
- VGLINEPATTERNCENTER

  The numerical value for this constant is 2.
- VGLINEPATTERNDASH

  The numerical value for this constant is 1.
- VGLINEPATTERNDASHDOT

  The numerical value for this constant is 6.
- VGLINEPATTERNDOT

  The numerical value for this constant is 5.
- VGLINEPATTERNMEDDASH

  The numerical value for this constant is 7.
- VGLINEPATTERNPHANTOM

  The numerical value for this constant is 3.
- VGLINEPATTERNSOLID

  The numerical value for this constant is 0.

VdLineStyle Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies the style used for lines.

Usage

VdLineStyle.*Constant*

Arguments

- VDLINE_BIGDASH

  The numerical value for this constant is 4.
- VDLINE_CENTER

  The numerical value for this constant is 2.
- VDLINE_DASH

  The numerical value for this constant is 1.
- VDLINE_DASHDOT

  The numerical value for this constant is 6.
- VDLINE_DOT

  The numerical value for this constant is 5.
- VDLINE_MEDDASH

  The numerical value for this constant is 7.
- VDLINE_PHANTOM

  The numerical value for this constant is 3.
- VDLINE_SOLID

  The numerical value for this constant is 0.

VdNameType Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies the format that is displayed for net names.

Usage

VdNameType.*Constant*

Arguments

- FULL_PATH_FROM_BLOCK

  The numerical value for this constant is 2.
- FULL_PATH_NAME

  The numerical value for this constant is 0.
- SHORT_NAME

  The numerical value for this constant is 1.

VdNotifyFlag Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies the events that trigger flags.

Usage

VdNotifyFlag.*Constant*

Arguments

- VDCN_APP_QUIT_AFTER

  The numerical value for this constant is 131072 (&H20000).
- VDCN_APP_QUIT_BEFORE

  The numerical value for this constant is 65536 (&H10000).
- VDCN_CREATE_AFTER

  The numerical value for this constant is 4.
- VDCN_CREATE_BEFORE

  The numerical value for this constant is 2.
- VDCN_DOC_CLOSE_AFTER

  The numerical value for this constant is 32768 (&H8000).
- VDCN_DOC_CLOSE_BEFORE

  The numerical value for this constant is 16384 (&H4000).
- VDCN_LOAD_AFTER

  The numerical value for this constant is 64 (&H40).
- VDCN_LOAD_BEFORE

  The numerical value for this constant is 32 (&H20).
- VDCN_OBJ_COPY

  The numerical value for this constant is 262144 (&H40000).
- VDCN_OBJ_CUT

  The numerical value for this constant is 524288 (&H80000).
- VDCN_ONACTIVATE

  The numerical value for this constant is 128 (&H80).
- VDCN_ONBUFFERPASTE

  The numerical value for this constant is 1024 (&H400).
- VDCN_ONDELETE

  The numerical value for this constant is 512 (&H200).

- VDCN_SAVE_AFTER

  The numerical value for this constant is 2048 (&H800).
- VDCN_SAVE_BEFORE

  The numerical value for this constant is 256 (&H100).
- VDCN_SAVEAS_AFTER

  The numerical value for this constant is 8192 (&H2000).
- VDCN_SAVEAS_BEFORE

  The numerical value for this constant is 4096 (&H1000).
- VDCN_SELECT

  The numerical value for this constant is 1.
- VDCN_UPDATE

  The numerical value for this constant is 8.

VdObjectClass Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies an object class.

Usage

VdObjectClass.*Constant*

Arguments

- ClassBOARD

  Printed circuit board file. The numerical value for this constant is 11 (&HB).
- ClassDLY

  Delay file. The numerical value for this constant is 10 (&HA).
- ClassPIN

  The numerical value for this constant is 3.
- ClassPKT

  Generic library file (/pkt). The numerical value for this constant is 5.
- ClassSCH_PAGE

  Schematic page (/sch). The numerical value for this constant is 0.
- ClassSPICE_MODEL

  SPICE model. The numerical value for this constant is 12 (&HC).
- ClassSYMBOL

  Symbol. The numerical value for this constant is 1.
- ClassVHDL

  VHDL source file (/vhdl). The numerical value for this constant is 6.
- ClassWIRELIST

  The numerical value for this constant is 2.
- ClassWVCWDFILE

  The numerical value for this constant is 7.
- ClassWVFILE

  The numerical value for this constant is 9.
- ClassWVWDIRFILE

  The numerical value for this constant is 8.

VdObjectType Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies an object type.

Usage

VdObjectType.*Constant*

Arguments

- VDTL_ANNOTATION

  Annotation layer. The numerical value for this constant is 1036 (&H40C).
- VDTL_BACKGROUND

  Background layer. The numerical value for this constant is 1038 (&H40E).
- VDTL_BORDER

  Border layer. The numerical value for this constant is 1040 (&H410).
- VDTL_HIGHLIGHT

  Highlight layer. The numerical value for this constant is 1041 (&H411).
- VDTL_SELECTION

  Selection layer. The numerical value for this constant is 1037 (&H40D).
- VDTL_VALUE

  Point. The numerical value for this constant is 1039 (&H40F).
- VDTS_ARC

  Arc. The numerical value for this constant is 4.
- VDTS_ATTRIBUTE

  Attribute. The numerical value for this constant is 6.
- VDTS_BLOCK

  Block. The numerical value for this constant is 11.
- VDTS_BOX

  Box. The numerical value for this constant is 1.
- VDTS_CIRCLE

  Circle. The numerical value for this constant is 3.
- VDTS_COMPONENT

  Component. The numerical value for this constant is 7.
- VDTS_FRAMEBOARD

  Frameboard. The numerical value for this constant is 19 (&H13).

- VDTS_LABEL

  Design. The numerical value for this constant is 8.
- VDTS_LINE

  Line. The numerical value for this constant is 0.
- VDTS_NET

  Net. The numerical value for this constant is 5.
- VDTS_PIN

  Pin. The numerical value for this constant is 9.
- VDTS_TEXT

  Text. The numerical value for this constant is 2.

VdObjectTypeMask Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies an object type mask (usually for selection purposes).

Usage

VdObjectTypeMask.*Constant*

Arguments

- VDM_ALL

  All objects. The numerical value for this constant is 16384 (&H4000).
- VDM_ARC

  Arc. The numerical value for this constant is 16 (&H10).
- VDM_ATTR

  Attribute. The numerical value for this constant is 64 (&H40).
- VDM_BLOCK

  Block. The numerical value for this constant is 2048 (&H800).
- VDM_BOX

  Box. The numerical value for this constant is 2.
- VDM_CIRCLE

  Circle. The numerical value for this constant is 8.
- VDM_COMP

  Component. The numerical value for this constant is 128 (&H80).
- VDM_COMPPIN

  Component pin. The numerical value for this constant is 4096 (&H1000).
- VDM_FRAMEBOARD

  Frameboard. The numerical value for this constant is 524288 (&H80000).
- VDM_LABEL

  Label. The numerical value for this constant is 256 (&H100).
- VDM_LINE

  Line or polygon. The numerical value for this constant is 1.
- VDM_NET

  Net. The numerical value for this constant is 32 (&H20).
- VDM_PIN

  Pin. The numerical value for this constant is 512 (&H200).

- VDM_SEGMENT

  Segment. The numerical value for this constant is 8192 (&H2000).
- VDM_TEXT

  Text. The numerical value for this constant is 4.

VdOnOff Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies mode settings.

Usage

VdOnOff.*Constant*

Arguments

- VDMD_OFF

  Value off. The numerical value for this constant is 0.
- VDMD_ON

  Value on. The numerical value for this constant is 1.

VdOpenMode Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies the mode for an opened block.

Usage

VdOpenMode.*Constant*

Arguments

- VDM_READ_LOCK

  Read-only (locked). The numerical value for this constant is 1.
- VDM_READ_ONLY

  Read-only (no lock). The numerical value for this constant is 2.
- VDM_READ_WRITE

  Read/write. The numerical value for this constant is 0.

VdOrientation Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies an orientation style.

Usage

VdOrientation.*Constant*

Arguments

- VDORIENT_IDENTITY

  Identity matrix index. The numerical value for this constant is 0.
- VDORIENT_MX

  Mirror on the x-axis (MX). The numerical value for this constant is 4.
- VDORIENT_MX180

  180 degree rotation on the x-axis. The numerical value for this constant is 6.
- VDORIENT_MX270

  270 degree rotation on the x-axis. The numerical value for this constant is 7.
- VDORIENT_MX90

  90 degree rotation on the x-axis. The numerical value for this constant is 5.
- VDORIENT_ROT180

  180 degree rotation index. The numerical value for this constant is 2.
- VDORIENT_ROT270

  270 degree rotation index. The numerical value for this constant is 3.
- VDORIENT_ROT90

  90 degree rotation index. The numerical value for this constant is 1.

VdOrigin Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies a point of origin.

Usage

VdOrigin.*Constant*

Arguments

- VDALIGN_LC

  Lower centered. The numerical value for this constant is 6.
- VDALIGN_LL

  Lower left. The numerical value for this constant is 3.
- VDALIGN_LR

  Lower right. The numerical value for this constant is 9.
- VDALIGN_MC

  Middle centered. The numerical value for this constant is 5.
- VDALIGN_ML

  Middle left. The numerical value for this constant is 2.
- VDALIGN_MR

  Middle right. The numerical value for this constant is 8.
- VDALIGN_NONE

  No origin point. The numerical value for this constant is 0.
- VDALIGN_UC

  Upper centered. The numerical value for this constant is 4.
- VDALIGN_UL

  Upper left. The numerical value for this constant is 1.
- VDALIGN_UR

  Upper right. The numerical value for this constant is 7.

VdParamMode Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies various modes for the Xpedition Designer session.

Usage

VdParamMode.*Constant*

Arguments

- VDMD_ABSOLUTE_OATS

  Use absolute OATs. The numerical value for this constant is 52 (&H34).
- VDMD_ARROWHEADS

  Enable symbol pin arrowheads. The numerical value for this constant is 40 (&H28).
- VDMD_AUTOPAN

  Enable the autopan feature for viewing. The numerical value for this constant is 26 (&H1A).
- VDMD_AUTO_RIPPER_BUS_SEGMENT

  The numerical value for this constant is 30 (&H1E).
- VDMD_AUTO_TEXT_ORIEN

  The numerical value for this constant is 29 (&H1D).
- VDMD_BELL

  Enable the warning bell. The numerical value for this constant is 5.
- VDMD_BIGCROSS

  Use the full-extent crosshair cursor. The numerical value for this constant is 48 (&H30).
- VDMD_BORDER_ON

  Enable the border for pages. The numerical value for this constant is 1.
- VDMD_BUS

  The numerical value for this constant is 0.
- VDMD_CABLING

  The numerical value for this constant is 61(&H3D).
- VDMD_CHECK_COMPDATES

  Enable the check component update mode. The numerical value for this constant is 46 (&H2E).
- VDMD_COARSE_GRID

  Enable coarse grid mode. The numerical value for this constant is 19 (&H13).
- VDMD_COMPDEF_ON

  The numerical value for this constant is 16 (&H10).

- VDMD_COMPTEXT_ON

  Enable component text mode. The numerical value for this constant is 4.
- VDMD_CONSTRAINT

  Enable constraint mode. The numerical value for this constant is 54 (&H36).
- VDMD_CONTEXT_WINDOW

  The numerical value for this constant is 10.
- VDMD_DB_ERR_VERBOSE

  Enable verbose error messages. The numerical value for this constant is 23 (&H17).
- VDMD_DBOX_ON

  The numerical value for this constant is 17 (H&11).
- VDMD_DEF_USESHEET1

  The numerical value for this constant is 44 (H&2C).
- VDMD_DETAIL

  The numerical value for this constant is 6.
- VDMD_DYNAMIC_PANNING

  The numerical value for this constant is 57 (H&39).
- VDMD_DYNAMIC_PLOTSIZE

  The numerical value for this constant is 36 (H&24).
- VDMD_DYNAMIC_XY

  The numerical value for this constant is 20 (H&14).
- VDMD_EURODATE_ON

  The numerical value for this constant is 51 (H&33).
- VDMD_EXCLUDE_GLOBALS_FM_UNIQUE_ON_COPY

  The numerical value for this constant is 55 (H&37).
- VDMD_EXPEDITION_ZOOM

  The numerical value for this constant is 59 (H&3B).
- VDMD_FUB_PINTYPE_ON

  The numerical value for this constant is 25 (H&19).
- VDMD_FULL_VHDL_CHECKS

  The numerical value for this constant is 28 (H&1C).
- VDMD_GRID_ON

  The numerical value for this constant is 2.
- VDMD_HEADER_ON

  The numerical value for this constant is 3.
- VDMD_LABELBRACKETS

  The numerical value for this constant is 50 (H&32).

- VDMD_MAPTOBLACK

  The numerical value for this constant is 27 (H&1B).
- VDMD_MIDSTROKE

  The numerical value for this constant is 58 (H&3A).
- VDMD_NAME

  The numerical value for this constant is 12.
- VDMD_NETS_IN_SPACE

  The numerical value for this constant is 21 (H&15).
- VDMD_NON_UNDOABLE_MOVE

  The numerical value for this constant is 39 (H&27).
- VDMD_OATCHECK

  The numerical value for this constant is 49 (H&31).
- VDMD_OATS

  The numerical value for this constant is 22 (H&16).
- VDMD_OLD_UNLIMITED_TEXT

  The numerical value for this constant is 34 (H&22).
- VDMD_OTO_BYPASS

  The numerical value for this constant is 35 (H&23).
- VDMD_PIN_TOOLTIPS

  The numerical value for this constant is 47 (H&2F).
- VDMD_PLACEHOLDER

  The numerical value for this constant is 32 (H&20).
- VDMD_PNUMS_ON

  The numerical value for this constant is 13.
- VDMD_PRESERVE_CASE

  The numerical value for this constant is 37 (H&25).
- VDMD_PROJECT_PLOT_ON_PC

  The numerical value for this constant is 53 (H&35).
- VDMD_RIPPERS_ON

  The numerical value for this constant is 901 (H&385).
- VDMD_RNUMS_ON

  The numerical value for this constant is 15.
- VDMD_SCHEMATIC_TABS

  The numerical value for this constant is 60 (H&3C).
- VDMD_SELNAME_ON

  The numerical value for this constant is 24 (H&18).

- VDMD_SNAP_PIN

  The numerical value for this constant is 7.
- VDMD_SORT_ON

  The numerical value for this constant is 14.
- VDMD_STROKES

  The numerical value for this constant is 56 (H&38).
- VDMD_THICKNETS_ON

  The numerical value for this constant is 900 (H&384).
- VDMD_UNDO

  The numerical value for this constant is 11.
- VDMD_UNIQUE_LABEL

  The numerical value for this constant is 8.
- VDMD_UNLIMITED_TEXT

  The numerical value for this constant is 31 (H&1F).
- VDMD_VALUES_ON

  The numerical value for this constant is 9.
- VDMD_WIR_CONT_CHAR

  The numerical value for this constant is 33 (H&21).
- VDMD_XTRAERRS_ON

  The numerical value for this constant is 18 (H&12).

VdParamValue Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies various values used in the Xpedition Designer session.

Usage

VdParamValue.*Constant*

Arguments

- VDVAL_ADISTANCE

  Avoidance distance. The numerical value for this constant is 10 (&HA).
- VDVAL_ANNO_SIZE

  Default annotation size. The numerical value for this constant is 16 (&H10).
- VDVAL_ATTR_ON_SPLIT

  Default placement of attributes on net split. The numerical value for this constant is 29 (&H1D).
- VDVAL_BLOCK_TYPE

  The default block type. The numerical value for this constant is 15 (&HF).
- VDVAL_BOX_SIZE

  Default dangle box size. The numerical value for this constant is 12 (&HC).
- VDVAL_BUBBLE_SIZE

  Bubble size. The numerical value for this constant is 8.
- VDVAL_BUS_DOT_SIZE

  Bus dot size. The numerical value for this constant is 17 (&H11).
- VDVAL_DEFMETHOD

  Default borders method. The numerical value for this constant is 22 (&H16).
- VDVAL_DOT_SIZE

  Dot size. The numerical value for this constant is 1.
- VDVAL_DOTSIZE_THREE_SEGMENT

  Default size for bus dots on 3-segment joints. The numerical value for this constant is 28 (&H1C).
- VDVAL_GRID

  Grid spacing. The numerical value for this constant is 0.
- VDVAL_GRID_HIGHLIGHT_INTERVAL

  Default interval for grid highlight. The numerical value for this constant is 26 (&H1A).
- VDVAL_LABEL_ON_SPLIT

Default placement of label on net split. The numerical value for this constant is 30 (&H1E).

- VDVAL_LABELTHRESHOLD

  Number of labels needed to trigger a sub-menu. The numerical value for this constant is 24 (&H18).

- VDVAL_LONG_LINE_ERRORS

  Report long line errors. The numerical value for this constant is 20 (&H14).

- VDVAL_MRU_SIZE

  The numerical value for this constant is 31 (&H1F).

- VDVAL_NET_LENGTH

  Default length for net stubs. The numerical value for this constant is 27 (&H1B).

- VDVAL_NET_SPACING

  Default spacing for Auto Net Array. The numerical value for this constant is 25 (&H19).

- VDVAL_NEW_ATTR_VIS

  New attribute visibility. The numerical value for this constant is 14 (&HE).

- VDVAL_NO_UNDO_CBA_MOVE

  Report “no undo connect by abutment” moves. The numerical value for this constant is 21 (&H15).

- VDVAL_ORIENTATION

  Default drawing orientation. The numerical value for this constant is 23 (&H17).

- VDVAL_ROUTE_MODE

  Route mode. The numerical value for this constant is 9.

- VDVAL_SCOPE

  Default label scope. The numerical value for this constant is 11 (&HB).

- VDVAL_SELECTION

  Default selection distance. The numerical value for this constant is 7.

- VDVAL_SHEET_SIZE

  The numerical value for this constant is 6.

- VDVAL_STROKE_DELAY

  The numerical value for this constant is 32 (&H20).

- VDVAL_TEXT_THRESHOLD

  The numerical value for this constant is 13.

- VDVAL_TORIGIN

  The numerical value for this constant is 4.

- VDVAL_TSIZE

  The numerical value for this constant is 3.

- VDVAL_UNDO

  The numerical value for this constant is 18 (&H12).

- VDVAL_WIDTH

  The numerical value for this constant is 5.

VdPEFlowMode Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies the PE flow mode.

Usage

VdPEFlowMode.*Constant*

Arguments

- VDPE_NONE

  The numerical value for this constant is 0.
- VDPE_BASE

  The numerical value for this constant is 1.
- VDPADSPRO_SE

  The numerical value for this constant is 2.

VdPinEndType Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies a pin end type.

Usage

VdPinEndType.*Constant*

Arguments

- VDPIN_BOUNDARY

  Boundary location (closest to bounding box). The numerical value for this constant is 1.
- VDPIN_INTERIOR

  Interior location (furthest from bounding box). The numerical value for this constant is 0.

VdRasterop Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies raster operations.

Usage

VdRasterop.*Constant*

Arguments

- VGMODEAND

  AND mode - Performs an AND of pixels. The numerical value for this constant is 7.
- VGMODENAND

  NAND mode - Performs a NAND of pixels. The numerical value for this constant is 3.
- VGMODENOR

  NOR mode - Performs a NOR of pixels. The numerical value for this constant is 5.
- VGMODENREP

  NOT mode - Performs a NOT of pixels; that is, pixels are not replaced. The numerical value for this constant is 4.
- VGMODENXOR

  XNOR (equivalence) mode - Performs an XNOR of pixels. The numerical value for this constant is 6.
- VGMODEOR

  OR mode - Performs an OR of pixels. The numerical value for this constant is 1.
- VGMODEREPLACE

  Replace mode - Overwrites pixels. The numerical value for this constant is 0.
- VGMODEXOR

  XOR mode - Performs an XOR of pixels. The numerical value for this constant is 2.

VdScope Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies the scope.

Usage

VdScope.*Constant*

Arguments

- VDGLOBAL_SCOPE

  The scope is global (across the entire design). The numerical value for this constant is 1.
- VDLOCAL_SCOPE

  The scope is local to the block. The numerical value for this constant is 0.

VdSegmentEndType Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies the segment end type.

Usage

VdSegmentEndType.*Constant*

Arguments

- VDSEG_PTHIGH

  Low segment. The numerical value for this constant is 1.
- VDSEG_PTLOW

  High segment. The numerical value for this constant is 0.

VdSelectionType Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies the selection notification types.

Usage

VdSelectionType.*Constant*

Arguments

- VDSELECT_NOTIFY

  Selection has occurred. The numerical value for this constant is 0.
- VDSELECT_NOTIFY_ADD

  Add selection has occurred. The numerical value for this constant is 1.
- VDSELECT_NOTIFY_REMOVE

  Selection has been cancelled. The numerical value for this constant is 2.

VdSense Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies the sense for labels and pins.

Usage

VdSense.*Constant*

Arguments

- VDINVERTED

  Inverted (draws an overbar or tilde). The numerical value for this constant is 1.
- VDNOTINVERTED

  Not inverted. The numerical value for this constant is 0.

VdSheetSize Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies sheet size.

Usage

VdSheetSize.*Constant*

Arguments

- VDSHEET_A0_SIZE

  Legacy metric A0 size drawing. The numerical value for this constant is 9.
- VDSHEET_A0L_SIZE

  Metric A0 size landscape drawing. The numerical value for this constant is 20 (&H14).
- VDSHEET_A0P_SIZE

  Metric A0 size portrait drawing. The numerical value for this constant is 30 (&H1E).
- VDSHEET_A1_SIZE

  Legacy metric A1 size drawing. The numerical value for this constant is 8.
- VDSHEET_A1L_SIZE

  Metric A1 size landscape drawing. The numerical value for this constant is 19 (&H13).
- VDSHEET_A1P_SIZE

  Metric A1 size portrait drawing. The numerical value for this constant is 29 (&H1D).
- VDSHEET_A2_SIZE

  Legacy metric A2 size drawing. The numerical value for this constant is 7.
- VDSHEET_A2L_SIZE

  Metric A2 size landscape drawing. The numerical value for this constant is 18 (&H12).
- VDSHEET_A2P_SIZE

  Metric A2 size portrait drawing. The numerical value for this constant is 28 (&H1C).
- VDSHEET_A3_SIZE

  Legacy metric A3 size drawing. The numerical value for this constant is 6.
- VDSHEET_A3L_SIZE

  Metric A3 size landscape drawing. The numerical value for this constant is 17 (&H11).
- VDSHEET_A3P_SIZE

  Metric A3 size portrait drawing. The numerical value for this constant is 27 (&H1B).
- VDSHEET_A4_SIZE

  Legacy metric A4 size drawing. The numerical value for this constant is 5.

- VDSHEET_A4L_SIZE

  Metric A4 size landscape drawing. The numerical value for this constant is 16 (&H10).
- VDSHEET_A4P_SIZE

  Metric A4 size portrait drawing. The numerical value for this constant is 26 (&H1A).
- VDSHEET_AL_SIZE

  An A size, landscape-oriented drawing. The numerical value for this constant is 11 (&HB).
- VDSHEET_AP_SIZE

  An A size, portrait-oriented drawing. The numerical value for this constant is 21 (&H15).
- VDSHEET_ASIZE

  Legacy A size drawing. The numerical value for this constant is 0.
- VDSHEET_BL_SIZE

  Legacy B size drawing (landscape). The numerical value for this constant is 12 (&HC).
- VDSHEET_BP_SIZE

  Legacy B size drawing (portrait). The numerical value for this constant is 22 (&H16).
- VDSHEET_BSIZE

  Legacy B size drawing. The numerical value for this constant is 1.
- VDSHEET_CL_SIZE

  Legacy C size drawing (landscape). The numerical value for this constant is 13 (&HD).
- VDSHEET_CP_SIZE

  Legacy C size drawing (portrait). The numerical value for this constant is 23 (&H17).
- VDSHEET_CSIZE

  Legacy C size drawing. The numerical value for this constant is 2.
- VDSHEET_DL_SIZE

  Legacy D size drawing (landscape). The numerical value for this constant is 14 (&HE).
- VDSHEET_DP_SIZE

  Legacy D size drawing (portrait). The numerical value for this constant is 24 (&H18).
- VDSHEET_DSIZE

  Legacy D size drawing. The numerical value for this constant is 3.
- VDSHEET_EL_SIZE

  Legacy E size drawing (landscape). The numerical value for this constant is 15.
- VDSHEET_EP_SIZE

  Legacy E size drawing (portrait). The numerical value for this constant is 25 (&H19).
- VDSHEET_ESIZE

  Legacy E size drawing. The numerical value for this constant is 4.
- VDSHEET_ZSIZE

  A custom size drawing. The numerical value for this constant is 10.

VdSide Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies the pin side.

Usage

VdSide.*Constant*

Arguments

- VDBOTTOM

  Bottom. The numerical value for this constant is 1.
- VDLEFT

  Left. The numerical value for this constant is 2.
- VDRIGHT

  Right. The numerical value for this constant is 3.
- VDTOP

  Top. The numerical value for this constant is 0.

VdSilentMode Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies silent mode severity settings.

Usage

VdSilentMode.*Constant*

Arguments

- VDSM_ALL

  All messages are silent. The numerical value for this constant is 1.

  Note:
  When SilentMode is set to VDSM_ALL, the user is not given the option to discard core dump files if a crash occurs. In that case, core dump files are automatically written to the WDIR directory.

- VDSM_NONE

  No messages are silent. The numerical value for this constant is 0.

VdSourceDocumentType Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies source document types.

Usage

VdSourceDocumentType.*Constant*

Arguments

- VDSOURCE_SPICE

  Spice source. The numerical value for this constant is 3.
- VDSOURCE_TEXT

  Text source. The numerical value for this constant is 0.
- VDSOURCE_VERILOG

  Verilog source. The numerical value for this constant is 2.
- VDSOURCE_VHDL

  VHDL-AMS source. The numerical value for this constant is 1.

VdSplineOrder Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies spline order.

Usage

VdSplineOrder.*Constant*

Arguments

- VGSPLINEROUND

  Spline curves loosely continue around control points. The numerical value for this constant is 5.

- VGSPLINESHARP

  Spline curves closely continue around control points. The numerical value for this constant is 3.

- VGSPLINESMOOTH

  Spline curves smoothly continue around control points. The numerical value for this constant is 4.

VdSplineType Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies spline type.

Usage

VdSplineType.*Constant*

Arguments

- VGSPLINECLOSED

  Specifies an open spline (last point does not connect to first point). The numerical value for this constant is 1.

- VGSPLINEOPEN

  Specifies a close spline (last point continues back to first point). The numerical value for this constant is 0.

VdSymbolType Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies symbol block type.

Usage

VdSymbolType.*Constant*

Arguments

- VDB_ANNOTATE

  Annotate (graphics). The numerical value for this constant is 3.
- VDB_COMPOSITE

  Composite (hierarchical). The numerical value for this constant is 0.
- VDB_MODULE

  Module (leaf). The numerical value for this constant is 1.
- VDB_PIN

  Pin. The numerical value for this constant is 4.

VdTextFlags Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies text flags.

Usage

VdTextFlags.*Constant*

Arguments

- vgTextMIRROR_FLAG

  Draw mirrored text. The numerical value for this constant is 32 (&H20).
- vgTextNORMAL

  Draw normal text. The numerical value for this constant is 0.
- vgTextOVERSCORE_FLAG

  Draw text with an overbar. The numerical value for this constant is 16 (&H10).
- vgTextUNDERSCORE_FLAG

  Draw text with an underscore. The numerical value for this constant is 64 (&H40).

VdUpdateOOScope Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Sets the scope of an update.

Usage

VdUpdateOOScope.*Constant*

Arguments

- VDUOO_BOARD

  Update board. The numerical value for this constant is 1.
- VDUOO_Project

  Update project. The numerical value for this constant is 0.
- VDUOO_SCHEMATIC

  Update schematic. The numerical value for this constant is 2.
- VDUOO_SHEET

  Update sheet. The numerical value for this constant is 3.

VdUpdateOtherObjects Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Determines which objects are updated.

Usage

VdUpdateOtherObjects.*Constant*

Arguments

- VDUOO_BORDERS

  The numerical value for this constant is 32 (&H20).
- VDUOO_BUSRIPPERS

  The numerical value for this constant is 4.
- VDUOO_BUSSIGNALS

  The numerical value for this constant is 2.
- VDUOO_PRINT_ORDER

  The numerical value for this constant is 16 (&H10).
- VDUOO_CROSS_REFERENCE

  The numerical value for this constant is 8.
- VDUOO_PROPERTIES

  The numerical value for this constant is 1.

Description

The constant is a binary flag and can represent one or more objects. You can use a bitwise OR operation to combine arguments. For example:

```
31 (11111) = All objects are enabled

 5 (00101) = BUSRIPPERS and PROPERTIES are enabled

 2 (00010) = BUSSIGNALS is enabled
```

VdVisibilityFlag Enum

Scope: Xpedition Designer schematic editor

Prerequisite: None

Specifies attribute visibility flags.

Usage

VdVisibilityFlag.*Constant*

Arguments

- VDINVISIBLE

  Invisible (not shown at all). The numerical value for this constant is 0.
- VDNAMEVISIBLE

  Only the name of the attribute is visible. The numerical value for this constant is 2.
- VDSAMEVISIBLE

  The numerical value for this constant is 4.
- VDVALUEVISIBLE

  Only the value of attribute is visible. The numerical value for this constant is 3.
- VDVISIBLE

  All elements (name and value) are visible. The numerical value for this constant is 1.

VdWhichJoint Enum

Scope: Xpedition Designer schematic editor

Prerequisites: None.

Specifies joint end points.

Usage

VdWhichJoint.*Constant*

Arguments

- VDJ_HIGH

  Specifies the high joint (highest coordinates). The numerical value for this constant is 1.
- VDJ_LOW

  Specifies the low joint (lowest coordinates). The numerical value for this constant is 0.

WindowStyles Enum

Scope: Schematic editor

Prerequisites: None

Specifies the command window style.

Usage

WindowStyles.*Constant*

Arguments

- **RedirectOutput**

  The numerical value for this constant is 2.
- **WindowInvisible**

  The numerical value for this constant is 0.
- **WindowVisible**

  The numerical value for this constant is 1.

Chapter 6
Scripting with Databook

You can create scripts to automate Databook tasks, such as controlling design objects, menus, and system responses. When you create scripts, Databook components become objects that you control programmatically.

That is, the scripts are event handlers and executing a certain event causes Databook to call the script.

The Databook object model consists of objects which have properties and Members.

Figure 6. Databook Object Model

Note:
Databook scripting does not support the C++ programming language or dual interfaces for an object.

Databook Objects
Configuring the Databook Script

Databook Objects

To create a script to control Databook objects, you need to know which objects to use and the relationship between the objects. These objects are Databook components converted to programmatic objects for use in scripts.

Databook is limited to event processing. An event is a user-defined action or occurrence to which a Databook script responds. All Databook scripting events are enabled by default.

Table 43. Databook Objects

Object	Description
Attribute Object	You can add an Attribute to the Databook object name.
Attributes Object	You can add a collection of Attributes to the Databook object name.
Component Object	You can add a Component to the Databook object name.
Components Object	You can add a collection of Components to a Databook object name. You can assign Count and Item properties to the Components object.
Property Object	You can add a Property to the Databook object name. The Property object is a collection of properties.

Attribute Object

You can add an Attribute to the Databook object name.

Table 44. Attribute Object Properties

Property	Description
Name Property (Attribute Object)	Sets or returns the name of the attribute.
NameVisible Property (Attribute Object)	Sets or returns the visibility of the attribute name.
Value Property (Attribute Object)	Sets or returns the value of the attribute.
ValueVisible Property (Attribute Object)	Sets or returns the visibility of the attribute value.

Name Property (Attribute Object)

Object: Attribute Object

Access: Read/Write

Prerequisites: None

Sets or returns the name of the attribute.

Usage

Attribute.**Name**

Arguments

None

Return Values

String. A string that represents the name of the attribute.

Examples

```
For Each attr In CompObj.Attributes
   sAttrName="Attribute name: " & attr.Name
   MsgBox sAttrName
Next
```

NameVisible Property (Attribute Object)

Object: Attribute Object

Access: Read/Write

Prerequisites: None

Sets or returns the visibility of the attribute name.

Usage

Attribute.**NameVisible**

Arguments

None

Return Values

Boolean. A Boolean value that represents the visibility of the attribute name. A value of True indicates that the attribute name is visible; False indicates that the attribute name is not visible.

Examples

```
' set the visibility of an attribute name
set attr = component.Attributes.Item("Value")
attr.NameVisible = FALSE
```

Value Property (Attribute Object)

Object: Attribute Object

Access: Read/Write

Prerequisites: None

Sets or returns the value of the attribute.

Usage

Attribute.**Value**

Arguments

None

Return Values

String. A string that represents the value of the attribute.

Examples

```
For Each attr In CompObj.Attributes
   sCmpData="Attribute name: " & attr.Name & ", Value: " & attr.Value
   MsgBox sCmpData
Next
```

ValueVisible Property (Attribute Object)

Object: Attribute Object

Access: Read/Write

Prerequisites: None

Sets or returns the visibility of the attribute value.

Usage

Attribute.**ValueVisible**

Arguments

None

Return Values

Boolean. A Boolean value that represents the visibility of the attribute value. A value of True indicates that the attribute value is visible; False indicates that the attribute value is not visible.

Examples

```
' set the visibility of an attribute value
set attr = component.Attributes.Item("Tolerance")
attr.ValueVisible = FALSE
```

Attributes Object

You can add a collection of Attributes to the Databook object name.

Table 45. Attributes Object Methods and Properties

Method or Property	Description
Add Method (Attributes Object)	Adds an attribute to the Attributes object.
Count Property (Attributes Object)	Returns the number of attribute objects in the Attributes object.

Add Method (Attributes Object)

Object: Attributes Object

Prerequisites: None

Adds an attribute to the Attributes object.

Note:
The isOat argument for this method is deprecated; however, it is still required. A runtime error will result if you do not provide the argument.

Usage

Attributes.**Add** (ByVal *Name* As String, ByVal *Value* As String, ByVal *NameVisible* As Boolean, ByVal *ValueVisible* As Boolean, ByVal *isOat* As Boolean)

Arguments

- Name

  String. A string that represents the attribute object name.
- Value

  String. A string that represents the attribute object value.
- NameVisible

  Boolean. A Boolean value that represents the name visibility setting. If True, the attribute name is visible; if False, the attribute name is not visible.
- ValueVisible

  Boolean. A Boolean value that represents the value visibility setting. If True, the attribute value is visible; if False, the attribute value is not visible.
- isOat

  (Deprecated) A Boolean value that represents whether the attribute has an instance value (formerly referred to as an OAT). If True, the attribute has an instance value; if False, the attribute does not have an instance value. This argument is deprecated, but still required.

Examples

```
' Add a property
component.Attributes.Add "TEST", "SCRIPTING", TRUE, TRUE, TRUE
```

Count Property (Attributes Object)

Object: Attributes Object

Access: Read-Only

Prerequisites: None

Returns the number of attribute objects in the Attributes object.

Usage

Attributes.**Count**

Arguments

None

Return Values

Long. A long that represents the number of attributes in the collection.

Component Object

You can add a Component to the Databook object name.

Table 46. Component Object Properties

Property	Description
Attributes Property (Component Object)	Returns the collection of attributes for the component.
Instances Property (Component Object)	Returns all instance values on the component as an encoded string.
Library Property (Component Object)	Returns the library that is used in the Databook search for the component.
Properties Property (Component Object)	Returns a collection of one or more properties that are annotated to the Databook database object.
Symbol Property (Component Object)	Sets or returns the user-defined symbol name.

Attributes Property (Component Object)

Object: Component Object

Access: Read-Only

Prerequisites: None

Returns the collection of attributes for the component.

Usage

*Component.***Attributes**

Arguments

None

Return Values

Collection. A collection of attributes associated with the component.

Examples

```
attribs = CompObj.Attributes
```

Instances Property (Component Object)

Object: Component Object

Access: Read-Only

Prerequisites: None

Returns all instance values on the component as an encoded string.

Usage

Component.**Instances**

Arguments

None

Return Values

String. An encoded string containing instances and their values.

Examples

```
' List the instance name (UID) of the component after it is added
'
Set CompInst = component.Instances
For Each name In CompInsts
   MsgBox "InstanceName="& name,,"UID"
Next
```

Library Property (Component Object)

Object: Component Object

Access: Read-Only

Prerequisites: None

Returns the library that is used in the Databook search for the component.

Usage

Component.**Library**

Arguments

None

Return Values

String. A string that contains the full name of the library.

Examples

See Application_LoadComponent Event for an example that uses the Library property.

Properties Property (Component Object)

Object: Component Object

Access: Read-Only

Prerequisites: None

Returns a collection of one or more properties that are annotated to the Databook database object.

Usage

Component.**Properties**

Arguments

None

Return Values

Collection. A collection of properties that are annotated to the Databook database object. The collection contains one or more properties.

Examples

```
' Return all properties of the component
attribs = obj.Properties

' Return the "Name" property of the component
name = obj.Properties("Name")
```

Symbol Property (Component Object)

Object: Component Object

Access: Read/Write

Prerequisites: None

Sets or returns the user-defined symbol name.

Usage

Component.**Symbol**

Arguments

None

Return Values

String. A string that contains the user-defined name for the symbol.

Examples

```
MsgBox CompObj.Symbol
```

Components Object

You can add a collection of Components to a Databook object name. You can assign Count and Item properties to the Components object.

Table 47. Components Object Methods, Properties, and Events

Method, Property, or Event	Description
Count Property (Components Object)	Returns the number of component objects in the Components object.
Item Method (Components Object)	Returns a Component object contained in the collection.
Application_AddComponent Event	You can use the Application_AddComponent event to check a component's properties before adding it to the schematic. For example, checking if the approved part's properties exist before instantiating a component can help prevent adding unapproved parts to the schematic.
Application_AfterAddComponent Event	The Application_AfterAddComponent event is used in a Databook script to make direct changes to the component on a schematic.
Application_AfterAnnotateComponent Event	The Application_AfterAnnotateComponent event is used in a Databook script to make direct changes to the component on a schematic.
Application_AnnotateComponent Event	The Application_AnnotateComponent event is used in a Databook script to prevent unapproved parts from being added to the schematic by making sure the approved part property exists before instantiating a component.
Application_LoadComponent Event	The Application_LoadComponent event is used in a Databook script to override the mapping from the loaded component to the library used in the Databook search.

Count Property (Components Object)

Object:

Access: Read-Only

Prerequisites: None

Returns the number of component objects in the Components object.

Usage

Components.**Count**

Arguments

None

Return Values

Long. A long that represents the number of components in the collection.

Item Method (Components Object)

Object: Components Object

Prerequisites: None

Returns a Component object contained in the collection.

Usage

Components.**Item** (ByVal *arg* As Long), or

Components (*arg*)

Arguments

- arg

  A zero-based index of a component, or the name of a component.

Description

Applies to the Components object (default method). See Components Object for more information.

Application_AddComponent Event

Scope: Schematic editor

Object: Components Object

Prerequisites: None

You can use the Application_AddComponent event to check a component's properties before adding it to the schematic. For example, checking if the approved part's properties exist before instantiating a component can help prevent adding unapproved parts to the schematic.

Usage

Sub **Application_AddComponent** (ByVal *component* As Component)

Note:
Component.Continue = FALSE setting cancels the add component function.

Arguments

- component

  Component. The component to be added to the schematic.

Description

Applies to the Components object (see Components Object). The script calls the event before adding a component to the schematic.

Examples

This event handler sets the visibility of a property and removes a property from the collection:

```
Function Application_AddComponent(component)
'''
  'Add a property
    component.Attributes.Add "TEST", "SCRIPTING", TRUE, TRUE, TRUE
  'set the visibility of a property
   set attr = component.Attributes.Item("Value")
   attr.NameVisible = FALSE
  'Remove a property
   component.Attributes.Remove("Tolerance")
  'If the Continue property is TRUE, then the component will be
  'added to the schematic
  'If the Continue property is FALSE, then the component is not added
   Component.Continue = TRUE
End Function
```

Related Topics

Application_AfterAddComponent Event

Application_AfterAnnotateComponent Event

Application_AnnotateComponent Event

Application_LoadComponent Event

Application_SelectComponent Event

Application_ViewDocument Event

Application_AfterAddComponent Event

Scope: Schematic editor

Object: Components Object

Prerequisites: None

The Application_AfterAddComponent event is used in a Databook script to make direct changes to the component on a schematic.

Usage

Sub **Application_AfterAddComponent** (ByVal *component* As Component)

Arguments

- component

  Component. The component that was just added to the schematic.

Description

Applies to the Components object (see Components Object) and is called by the script after a generic or unique component is added to the schematic.

Note:
This event is not called when a symbol is dragged from the Databook “CL View” tab symbol preview window to the schematic.

Examples

This event handler lists the instance name of the component after it is added.

```
Function Application_AfterAddComponent(component)
   '''
   ''' Hook up to running Xpedition Designer
   Set view = Viewdraw.ActiveView
   ''' When adding, this list always has one element - but we can still
   ''' use For Each to get to it
      For Each name In component.Instances
         '' Select the component
         '' Not required, but is usefull to know how for other operations
         view.SelectByName name
         MsgBox "InstanceName=" & name,,"UID"
      Next
End Function
```

Related Topics

Application_AddComponent Event

Application_AfterAnnotateComponent Event

Application_AnnotateComponent Event

Application_LoadComponent Event

Application_SelectComponent Event

Application_ViewDocument Event

Application_AfterAnnotateComponent Event

Scope: Schematic editor

Object:

Prerequisites: None

The Application_AfterAnnotateComponent event is used in a Databook script to make direct changes to the component on a schematic.

Usage

Sub **Application_AfterAnnotateComponent** (ByVal *component* As Component)

Arguments

- component

  Component. The component that was just annotated in the schematic.

Description

Applies to the Components object (see Components Object) and is called by the script after annotating a component on the schematic.

Examples

This event handler resets the slot of a component after it is annotated and reports the UID and count.

```
Function Application_AfterAnnotateComponent(component)
   '''
   ''' Hook up to running Xpedition Designer
   Set view = Viewdraw.ActiveView
   ''' A loop is needed since more than once component might be
   ''' annotated
 Set CompInst = component.Instances
 ' Count example
 CompInstCount = CompInst.count
 MsgBox CompInstCount,,"Instance Count"
     For Each name In CompInst
       '' Select the component
       view.SelectByName name
       '' Reset slot and REFDES
       Viewdraw.ExecuteCommand "pdbslot 3 No"
       MsgBox "Reset slot for " & name,,"Change Slot Message"
     Next
End Function
```

Related Topics

Application_AddComponent Event

Application_AfterAddComponent Event

Application_AnnotateComponent Event

Application_LoadComponent Event

Application_SelectComponent Event

Application_ViewDocument Event

Application_AnnotateComponent Event

Scope: Schematic editor

Object: Components Object

Prerequisites: None

The Application_AnnotateComponent event is used in a Databook script to prevent unapproved parts from being added to the schematic by making sure the approved part property exists before instantiating a component.

Usage

Sub **Application_AnnotateComponent** (ByVal *component* As Component)

Arguments

- component

  Component. The component to be annotated.

Description

The Application_AnnotateComponent event has the following characteristics:

- Applies to the Components object (Components Object) and is called by the script before annotating a component on a schematic.

  To control whether or not Databook calls this event, in the Databook Search window select **Configure > Edit Configuration > Preferences** and check or uncheck the available annotation options in the Configure Dialog Box.

- Annotation settings in Databook Configure dialog box impact if Application_AnnotateComponent event is called by Databook.

Examples

This example displays a message box listing which properties will be annotated:

```
Function Application_AnnotateComponent(component)
   '''
   '''
   dim string
     For Each attr In component.Attributes
        sAttr = sAttr & CHR(10) & attr.Name & "=" & attr.Value
     next
   MsgBox "Properties being annotated are: " + CHR(10) + sAttr,,"Info"
   ''' If the Continue property is TRUE, the attributes will be
   ''  annotated to the component
   ''' If the Continue property is FALSE, the attributes will not be
   ''  annotated
   Component.Continue = TRUE
End Function
```

Related Topics

Application_AddComponent Event

Application_AfterAddComponent Event

Application_AfterAnnotateComponent Event

Application_LoadComponent Event

Application_SelectComponent Event

Application_ViewDocument Event

Application_LoadComponent Event

Scope: Schematic editor

Object: Components Object

Prerequisites: None

The Application_LoadComponent event is used in a Databook script to override the mapping from the loaded component to the library used in the Databook search.

Usage

Sub **Application_LoadComponent** (ByVal *components* As Components)

Arguments

- components

  Collection. The collection containing one or more loaded components.

Description

Applies to the Components object (see Components Object) and is called by the script when loading components from the schematic and before searching the Databook database for the components.

Note:
If you click the Databook Live Verification button, this event is called by Databook when loading components from the schematic.

Examples

This example displays a dialog box showing the loaded components:

```
Function Application_LoadComponent(components)
   ''
   ''
   dim sCmpData
   For Each component In components
     sCmpData= sCmpData & component.Name & CHR(10)
     sCmpData= sCmpData & " Symbol=" & component.Symbol & CHR(10)
     sCmpData= sCmpData & " Library=" & component.Library & CHR(10)
     Set attrColl = component.Attributes
      For Each attr In attrColl
        sCmpData= sCmpData & " " & attr.Name & "=" & attr.Value & CHR(10)
      Next
     sCmpData= sCmpData & CHR(10)
   next
  MsgBox "The loaded components are:" & CHR(10) & CHR(10) & sCmpData
End Function
```

Related Topics

Application_AddComponent Event

Application_AfterAddComponent Event

Application_AfterAnnotateComponent Event

Application_AnnotateComponent Event

Application_SelectComponent Event

Application_ViewDocument Event

Property Object

You can add a Property to the Databook object name. The Property object is a collection of properties.

Table 48. Property Object Methods and Events

Method or Event	Description
Item Method (Property Object)	Returns a Property contained in the Property object.
Remove Method (Property Object)	Removes a property from the collection, using an index.
Application_SelectComponent Event	The Application_SelectComponent event is used in a Databook script to update a third-party viewer when selecting different components.
Application_ViewDocument Event	The Application_ViewDocument event is used in a Databook script to dynamically build URLs to web pages.

Item Method (Property Object)

Object: Property Object

Prerequisites: None

Returns a Property contained in the Property object.

Usage

Property.**Item** (ByVal *arg* As Long), or

Property (arg)

Arguments

- arg

  A zero-based index of a property, or the name of a property.

Description

Applies to the Property object (default method). See Property Object for more information.

Remove Method (Property Object)

Object: Property Object

Prerequisites: None

Removes a property from the collection, using an index.

Usage

Property.**Remove** (ByVal *arg* As Long)

Arguments

- arg

  A zero-based index of a property, or the name of a property.

Description

Applies to the Properties object (see Property Object).

Examples

```
PropObjs.Remove(3)
```

Application_SelectComponent Event

Scope: Schematic editor

Object: Property Object

Prerequisites: None

The Application_SelectComponent event is used in a Databook script to update a third-party viewer when selecting different components.

Usage

Sub **Application_SelectComponent** (ByVal *arg* As Object)

Arguments

- arg

  A row of properties selected in the Component Search results window.

Description

Applies to the Properties object (see Property Object) and is called by the script when a row is selected in the Databook Component Search results window.

Examples

This example displays a message box listing the properties of the selected row:

```
Function Application_SelectComponent(Attributes)
   '
   '
   dim sAttData
   For Each attr In Attributes
      attrName = attr.Name
  attrValue = attr.Value
      sAttData= sAttData & CHR(10) & attrName & "=" & attrValue
   Next
   MsgBox "The selected properties are:" & CHR(10) & sAttData,,"Summary"
End Function
```

Related Topics

Application_AddComponent Event

Application_AfterAddComponent Event

Application_AfterAnnotateComponent Event

Application_AnnotateComponent Event

Application_LoadComponent Event

Application_ViewDocument Event

Application_ViewDocument Event

Scope: Schematic editor

Object: Property Object

The Application_ViewDocument event is used in a Databook script to dynamically build URLs to web pages.

Usage

Sub **Application_ViewDocument** (ByVal *arg1* As Object, ByVal *arg2* As Object)

Arguments

- arg1

  A row of properties selected in the Component Search results window.

- arg2

  A property containing the document you want to view.

Description

Applies to the Properties objects (see Property Object), and is called by the script when you click a Web page link and before the document viewer launches, so that it can build a URL.

Note:
Enumerated values are not automatically available in a script file, so you must specify the numeric values for this data type (for example, VDM_COMP = 128).

Examples

This example launches notepad to view the document specified in the Databook Document field:

Note:
Mentor Graphics recommends that you use COM versioning syntax in script examples that use GetObject and CreateObject. Without COM versioning, the script will access the last installation to which the release switcher pointed.

```
Function Application_ViewDocument(properties, document)
   '
   '
   set wsh = CreateObject("WScript.Shell")
   ' Searches path for notepad.exe
   wsh.Run("notepad.exe " & "C:\DataSheets\" & document.Value)
   ' set the document value to empty ("") to prevent Databook
   ' from launching the viewer
End Function
```

Note:
You can use directory pointers to point to a central datasheet repository, as shown in the example.

Related Topics

Application_AddComponent Event

Application_AfterAddComponent Event

Application_AfterAnnotateComponent Event

Application_AnnotateComponent Event

Application_LoadComponent Event

Application_SelectComponent Event

Configuring the Databook Script

You must configure your script before Databook can read the file.

Prerequisites

- Create a Databook script file.

Procedure

1. In Databook Search window, right-click and select **Configure > Scripting > Settings**.
2. In the Scripting dialog box:
   - If you know the script file name, type the name into the Filename field.
   - If you want to browse for a script file, click the browse button.
3. [Optional] To modify the script file in a text editor, click Edit.

   To reload the file, select **Configure > Scripting > Reload Script**.
4. In the Language section, select the programming language for your script.

   If you select the User-Defined option, type the programming language name into the ProgID field.
5. Click **OK**.

Results

You have configured your script file and Databook can read the script.

Related Topics

Databook Objects

Appendix A
Changes to Schematic Editor Automation

The automation layer of Xpedition Designer has undergone significant changes in response to the new paradigm regarding the underlying iCDB database.

In order, however, to maintain the validity of legacy scripts that may have been developed within the community of users, Mentor Graphics has taken steps to ensure that the changes do not disrupt the execution of any such scripts.

More specifically, certain automation items have been removed while others have been redirected to accommodate the latest changes to Xpedition Designer. Wherever possible, Mentor Graphics has provided “redirection” for the removed items; that is, any call to a removed item will be redirected to another functionally-equivalent item that conforms with the new database paradigm.

Note:
There are certain other objects for which redirection is not possible. In those cases, a call for the object in question will produce an error message. Mentor Graphics has attempted to keep the number of these objects to a minimum.

Changes to Objects
Changes to Enumerated Types

Changes to Objects

The new paradigm for libraries, database, and other aspects of Xpedition Designer have rendered changes to objects in the automation layer.

These are touched on in the following sections.

Removed Objects
Changes to the Application Object
Changes to the Block Object
Changes to the Arc Object
Changes to the Attribute Object
Changes to the Box Object
Changes to the Circle Object
Changes to the Component Object
Changes to the Connection Object
Changes to the Label Object
Changes to the Line Object
Changes to the Net Object
Changes to the Pin Object
Changes to the Ripper Object
Changes to the Text Object
Changes to the Viewport Object

Removed Objects

Some automation objects (including methods and properties associated with them) from previous versions of Xpedition Designer have been completely removed.

Note:
Any scripts that reference these objects will produce error messages when they are run.

- **Library object** — This object was made obsolete with the introduction of a central library paradigm in Xpedition Designer.

Changes to the Application Object

As a result of the removal of *.wir* files, changes to the library paradigm, and other changes to the database in Xpedition Designer, some Application Object items have been removed.

The items shown in Table 49: Changes to the Application Objecthave been removed from the Application Object.

Note:
If any of these methods or properties appear in a script, they will be ignored.

Table 49. Changes to the Application Object

Removed method or property	Reason for removal
AddLibrary method	The library paradigm has changed such that local libraries are no longer relevant. They have been replaced by a central library, which can be managed within the "SymbolPartitions Object" on page 554.
AddLibraryAndSaveIni method	The library paradigm has changed such that local libraries are no longer relevant. They have been replaced by a central library, which can be managed within the "SymbolPartitions Object" on page 554.
AfterDocumentSave event	Irrelevant now that schematics are immediately saved to the database.
AfterSaveAndCheck event	Irrelevant now that schematics are immediately saved to the database.
AfterSheetRead event	Irrelevant now that schematics are immediately saved to the database.
AfterSheetReRead event	Irrelevant now that schematics are immediately saved to the database.
AttributeCanHaveOatValue method	Instance values are always enabled in Xpedition Designer. Therefore, this method will always return a value of "True."
AttributeValueMustBeUpper method	Properties in Xpedition Designer are now case-preserving and case-insensitive. Therefor, this method will always return a value of "False."
BeforeDocumentSave event	Irrelevant now that schematics are immediately saved to the database.
BeforeProjectModified event	Replaced by BeforeProjectChanged Event (Application Object).
BeforeSaveAndCheck event	Irrelevant now that schematics are immediately saved to the database.
ClearAllLibraries method	The library paradigm has changed such that local libraries are no longer relevant. They have been replaced by a central library, which can be managed within the "Label Object" on page 385.
CnsFileString property	Constraints are now kept in iCDB database.
ConstraintsModeChanged event	Constraints are now kept in iCDB database.

Table 49. Changes to the Application Object (continued)

Removed method or property	Reason for removal
CurrentProject property	Irrelevant with the changes made to project management. You can now retrieve project data using the "GetProjectData Method (Application Object)" on page 78.
DeleteAttribute method	Irrelevant with the changes made to property assignment.
DeleteLibrary method	The library paradigm has changed such that local libraries are no longer relevant. They have been replaced by a central library, which can be managed within the "Label Object" on page 385.
DeleteLibraryAndSaveIni method	The library paradigm has changed such that local libraries are no longer relevant. They have been replaced by a central library, which can be managed within the "Label Object" on page 385.
DeletePackagedAttribute method	Irrelevant with the changes made to property assignment.
DeleteSheet method	Sheets are no longer identified by number; this method has been replaced by DeleteSheet Method (SchematicSheetDocuments Collection).
DocumentSave method	Irrelevant now that schematics are immediately saved to the database.
DocumentSaveAs method	Irrelevant now that schematics are immediately saved to the database.
GetAttributeValues method	Irrelevant with the changes made to property assignment.
GetLibraries method	The library paradigm has changed such that local libraries are no longer relevant. They have been replaced by a central library, which can be managed within the "Label Object" on page 385.
GetPackagedAttributeValues method	Irrelevant with the changes made to property assignment.
IsLibDeletable method	The library paradigm has changed such that local libraries are no longer relevant. They have been replaced by a central library, which can be managed within the "Label Object" on page 385.
KickViewBase method	Irrelevant with removal of *.wir* files.
MoveLibrary method	The library paradigm has changed such that local libraries are no longer relevant. They have been replaced by a central library, which can be managed within the "Label Object" on page 385.
MoveLibraryAndSaveIni method	The library paradigm has changed such that local libraries are no longer relevant. They have been replaced by a central library, which can be managed within the "Label Object" on page 385.

Table 49. Changes to the Application Object (continued)

Removed method or property	Reason for removal
OATAdded event	Obsolete. With the changes to the database, there is no distinction specific to OAT attributes.
ObjectColor property	No longer used to determine the color of a schematic object.
ParamAddListName method	Xpedition Designer no longer has functionality to control promotions of this kind.
ParamGetListNames method	Xpedition Designer no longer has functionality to control promotions of this kind.
PrimaryDirectory property	Obsolete. You can now derive the same information using "GetProjectData Method (Application Object)" on page 78.
Project property	Irrelevant with the changes made to project management.
ProjectModified event	Replaced by ProjectChanged Event (Application Object).
Projman property	Irrelevant with the changes made to project management.
ReadIni method	Irrelevant now that Xpedition Designer now determines configuration from files other than the *.ini* file.
RemoveProjectSettingTab method	Irrelevant with the changes made to project management.
SaveIni method	Irrelevant since Xpedition Designer now saves all configuration files immediately upon change.
SetAttributeValues method	Irrelevant with the changes made to property assignment.
SetEnvVariable method	Obsolete
SetPackagedAttribueValue method	Irrelevant with the changes made to property assignment.
SynchronizesViewBase method	Irrelevant with removal of *.wir* files.
ViewBaseSession method	Irrelevant with removal of *.wir* files.

Changes to the Block Object

As a result of the removal of *.wir* files, changes to the library paradigm, and other changes to the database in Xpedition Designer, some Block Object items have been removed.

The items shown in Table 50: Changes to the Block Object have been removed from the Block Object.

Note:
If any of these methods or properties appear in a script, they will be ignored.

Table 50. Changes to the Block Object

Removed method or property	Reason for removal
AddComponent method	Obsolete.
AddComponentMoveMode method	Obsolete.
AddComponentMoveModeEx method	Obsolete.
AddPin method	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.
AddPinAtLocation method	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.
Attributes property	Irrelevant with the changes made to property assignment.
GetPackagedName method	Obsolete.
LibraryName property	The library paradigm has changed such that local libraries are no longer relevant. They have been replaced by a central library, which can be managed within the "Label Object" on page 385.

Changes to the Arc Object

As a result of the removal of *.wir* files, changes to the library paradigm, and other changes to the database in Xpedition Designer, some Arc Object items have been removed.

The items shown in Table 51: Changes to the Arc Object have been removed from the Arc Object.

Note:
If any of these methods or properties appear in a script, they will be ignored.

Table 51. Changes to the Arc Object

Removed method or property	Reason for removal
Color property	Irrelevant with the removal of the VdColor enumerated type.

Changes to the Attribute Object

As a result of the removal of *.wir* files, changes to the library paradigm, and other changes to the database in Xpedition Designer, some Attribute Object items have been removed.

The items shown in Table 52: Changes to the Attribute Object have been removed from the Attribute Object.

Note:
If any of these methods or properties appear in a script, they will be ignored.

Table 52. Changes to the Attribute Object

Removed method or property	Reason for removal
Color property	Irrelevant with the removal of the VdColor enumerated type.

Changes to the Box Object

As a result of the removal of *.wir* files, changes to the library paradigm, and other changes to the database in Xpedition Designer, some Box Object items have been removed.

The items shown in Table 53: Changes to the Box Object have been removed from the Box Object.

Note:
If any of these methods or properties appear in a script, they will be ignored.

Table 53. Changes to the Box Object

Removed method or property	Reason for removal
Color property	Irrelevant with the removal of the VdColor enumerated type.

Changes to the Circle Object

As a result of the removal of *.wir* files, changes to the library paradigm, and other changes to the database in Xpedition Designer, some Circle Object items have been removed.

The items shown in Table 54: Changes to the Circle Object have been removed from the Circle Object.

Note:
If any of these methods or properties appear in a script, they will be ignored.

Table 54. Changes to the Circle Object

Removed method or property	Reason for removal
Color property	Irrelevant with the removal of the VdColor enumerated type.

Changes to the Component Object

As a result of changes to the database in Xpedition Designer, some Component Object items have been removed.

The items shown in Table 55: Changes to the Component Object have been removed from the Component Object.

Note:
If any of these methods or properties appear in a script, they will be ignored.

Table 55. Changes to the Component Object

Removed method or property	Reason for removal
AddBatchOats method	Obsolete. With the changes to the database, there is no distinction specific to OAT attributes.

Changes to the Connection Object

As a result of the removal of *.wir* files, changes to the library paradigm, and other changes to the database in Xpedition Designer, some Connection Object items have been removed.

The items shown in Table 56: Changes to the Connection Object have been removed from the Component Object.

Note:
If any of these methods or properties appear in a script, they will be ignored.

Table 56. Changes to the Connection Object

Removed method or property	Reason for removal
CompPin property	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.

Changes to the Label Object

As a result of the removal of *.wir* files, changes to the library paradigm, and other changes to the database in Xpedition Designer, some Label Object items have been removed.

The items shown in Table 57: Changes to the Label Object have been removed from the Label Object.

Note:
If any of these methods or properties appear in a script, they will be ignored.

Table 57. Changes to the Label Object

Removed method or property	Reason for removal
Color property	Irrelevant with the removal of the VdColor enumerated type.

Table 57. Changes to the Label Object (continued)

Removed method or property	Reason for removal
Scope property	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.

Changes to the Line Object

As a result of the removal of *.wir* files, changes to the library paradigm, and other changes to the database in Xpedition Designer, some Line Object items have been removed.

The items shown in Table 58: Changes to the Line Object have been removed from the Line Object.

Note:
If any of these methods or properties appear in a script, they will be ignored.

Table 58. Changes to the Line Object

Removed method or property	Reason for removal
Color property	Irrelevant with the removal of the VdColor enumerated type.
Scope property	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.

Changes to the Net Object

As a result of the removal of *.wir* files, changes to the library paradigm, and other changes to the database in Xpedition Designer, some Net Object items have been removed.

The items shown in Table 59: Changes to the Net Object have been removed from the Net Object.

Note:
If any of these methods or properties appear in a script, they will be ignored.

Table 59. Changes to the Net Object

Removed method or property	Reason for removal
AddOat method	Obsolete. With the changes to the database, there is no distinction specific to OAT attributes.
Color property	Irrelevant with the removal of the VdColor enumerated type.

Changes to the Pin Object

As a result of the removal of *.wir* files, changes to the library paradigm, and other changes to the database in Xpedition Designer, some Pin Object items have been removed.

The items shown in Table 60: Changes to the Pin Object have been removed from the Pin Object.

Note:
If any of these methods or properties appear in a script, they will be ignored.

Table 60. Changes to the Pin Object

Removed method or property	Reason for removal
AddAttribute method	Obsolete due to changes in the database.
Color property	Obsolete due to changes in the database.

Changes to the Ripper Object

As a result of the removal of *.wir* files, changes to the library paradigm, and other changes to the database in Xpedition Designer, some Ripper Object items have been removed.

The items shown in Table 61: Changes to the Ripper Object have been removed from the Ripper Object.

Note:
If any of these methods or properties appear in a script, they will be ignored.

Table 61. Changes to the Ripper Object

Removed method or property	Reason for removal
GetConnectedObject method	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.
GetConnectedObjects method	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.

Changes to the Text Object

As a result of the removal of *.wir* files, changes to the library paradigm, and other changes to the database in Xpedition Designer, some Text Object items have been removed.

The items shown in Table 62: Changes to the Text Object have been removed from the Text Object.

Note:
If any of these methods or properties appear in a script, they will be ignored.

Table 62. Changes to the Text Object

Removed method or property	Reason for removal
Color property	Obsolete due to changes in the database.

Changes to the Viewport Object

As a result of the removal of *.wir* files, changes to the library paradigm, and other changes to the database in Xpedition Designer, some Viewport Object items have been removed.

The items shown in Table 63: Changes to the Viewport Object have been removed from the Viewport Object.

Note:
If any of these methods or properties appear in a script, they will be ignored.

Table 63. Changes to the Viewport Object

Removed method or property	Reason for removal
Color property	Obsolete due to changes in the database.

Changes to Enumerated Types

The new paradigm for libraries, database, and other aspects of Xpedition Designer have rendered changes to enumerated types in the automation layer.

These are touched on in the following sections.

Removed Enumerated Types
Changes to the VdAppEventDispatchID Enumerated Type
Changes to the VdDocumentAccess Enumerated Type
Changes to the VdNotifyFlag Enumerated Type
Changes to the VdObjectClass Enumerated Type
Changes to the VdObjectType Enumerated Type
Changes to the VdObjectTypeMask Enumerated Type

Removed Enumerated Types

The following enumerated types from previous versions of Xpedition Designer have been completely removed.

Note:
Any scripts that reference these enumerated types will produce error messages when they are run.

- **VdAnnoType** — this enumerated type was made obsolete by changes to the database.
- **VdColor** — this enumerated type was made obsolete by changes to the database.
- **VdLibraryType** — this enumerated type was made obsolete by changes to the database.
- **VdProjectSettingTab** — this enumerated type was made obsolete by changes to the database.
- **VdPromoteType** — this enumerated type was made obsolete by changes to the database.
- **VdScope** — this enumerated type was made obsolete by changes to the database.

Changes to the VdAppEventDispatchID Enumerated Type

As a result of the removal of *.wir* files, changes to the library paradigm, and other changes to the database in Xpedition Designer, some vdAppEventDispatchID Enum items have been removed.

The items shown in Table 64: Changes to the VdAppEventDispatchID Enumerated Type have been removed from the VdAppEventDispatchID Enum.

Note:
If any of these appear in a script, they will be ignored.

Table 64. Changes to the VdAppEventDispatchID Enumerated Type

Removed constant	Reason for removal
AFTER_DOCUMENT_SAVE	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.
AFTER_SAVE_AND_CHECK	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.
AFTER_SHEET_REREAD	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.
BEFORE_DOCUMENT_SAVE	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.
BEFORE_PROJECT_MODIFIED	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.
BEFORE_SAVE_AND_CHECK	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.
CONSTRAINTS_MODE_CHANGED	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.
DOCUMENT_SAVE	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.
DOCUMENT_SAVEAS	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.
OAT_ADDED	Obsolete. With the changes to the database, there is no distinction specific to OAT attributes.
PROJECT_MODIFIED	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.
SOURCE_FILE_MODIFIED	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.
SOURCEDOCUMENT_SAVE	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.

Changes to the VdDocumentAccess Enumerated Type

As a result of the removal of *.wir* files, changes to the library paradigm, and other changes to the database in Xpedition Designer, some VdDocumentAccess Enum items have been removed.

The items shown in Table 65: Changes to the VdDocumentAccess Enumerated Type have been removed from the VdDocumentAccess Enum.

Note:
If any of these appear in a script, they will be ignored.

Table 65. Changes to the VdDocumentAccess Enumerated Type

Removed constant	Reason for removal
BLOCK_WRITABLE	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.
OAT_READ_ONLY	Obsolete. With the changes to the database, there is no distinction specific to OAT attributes.
OAT_WRITABLE	Obsolete. With the changes to the database, there is no distinction specific to OAT attributes.
SYM_READ_LOCK	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.
SYM_READ_ONLY	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.
SYM_WRITABLE	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.
WIR_READ_ONLY	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.
WIR_WRITABLE	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.

Changes to the VdNotifyFlag Enumerated Type

As a result of the removal of *.wir* files, changes to the library paradigm, and other changes to the database in Xpedition Designer, some VdNotifyFlag Enum items have been removed.

The items shown in Table 66: Changes to the VdNotifyFlag Enumerated Type have been removed from the VdNotifyFlag Enum.

Note:
If any of these appear in a script, they will be ignored.

Table 66. Changes to the VdNotifyFlag Enumerated Type

Removed constant	Reason for removal
DOC_SAVE_AFTER	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.

Table 66. Changes to the VdNotifyFlag Enumerated Type (continued)

Removed constant	Reason for removal
DOC_SAVE_BEFORE	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.
DOC_SAVEAS_AFTER	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.
DOC_SAVEAS_BEFORE	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.

Changes to the VdObjectClass Enumerated Type

As a result of the removal of *.wir* files, changes to the library paradigm, and other changes to the database in Xpedition Designer, some VdObjectClass Enum items have been removed.

The items shown in Table 67: Changes to the VdObjectClass Enumerated Type have been removed from the VdObjectClass Enum.

Note:
If any of these appear in a script, they will be ignored.

Table 67. Changes to the VdObjectClass Enumerated Type

Removed constant	Reason for removal
OAT	Obsolete. Obsolete. With the changes to the database, there is no distinction specific to OAT attributes.
PIN	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.
SYMBOL	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.
WIRELIST	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.

Changes to the VdObjectType Enumerated Type

As a result of the removal of *.wir* files, changes to the library paradigm, and other changes to the database in Xpedition Designer, some VdObjectType Enum items have been removed.

The items shown in Table 68: Changes to the VdObjectType Enumerated Type have been removed from the VdObjectType Enum.

Note:
If any of these appear in a script, they will be ignored.

Table 68. Changes to the VdObjectType Enumerated Type

Removed constant	Reason for removal
LIBRARY	Irrelevant with the removal of *.wir* and *.sch* files and changes to the database paradigm.
OATATTRIBUTE	Obsolete. With the changes to the database, there is no distinction specific to OAT attributes.

Changes to the VdObjectTypeMask Enumerated Type

As a result of the removal of *.wir* files, changes to the library paradigm, and other changes to the database in Xpedition Designer, some VdObjectTypeMask Enum items have been removed.

The items shown in Table 68: Changes to the VdObjectType Enumerated Type have been removed from the VdObjectTypeMask Enum.

Note:
If any of these appear in a script, they will be ignored.

Table 69. Changes to the VdObjectTypeMask Enumerated Type

Removed constant	Reason for removal
OAT	Obsolete. With the changes to the database, there is no distinction specific to OAT attributes.

Third-Party Information

Details on open source and third-party software that may be included with this product are available in the *<your_software_installation_location>/legal* directory.