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

The terms and conditions governing the sale and licensing of Mentor Graphics products are set forth in written agreements between Mentor Graphics and its customers. No representation or other affirmation of fact contained in this publication shall be deemed to be a warranty or give rise to any liability of Mentor Graphics whatsoever.

MENTOR GRAPHICS MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

MENTOR GRAPHICS SHALL NOT BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS PUBLICATION OR THE INFORMATION CONTAINED IN IT, EVEN IF MENTOR GRAPHICS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

U.S. GOVERNMENT LICENSE RIGHTS: The software and documentation were developed entirely at private expense and are commercial computer software and commercial computer software documentation within the meaning of the applicable acquisition regulations. Accordingly, pursuant to FAR 48 CFR 12.212 and DFARS 48 CFR 227.7202, use, duplication and disclosure by or for the U.S. Government or a U.S. Government subcontractor is subject solely to the terms and conditions set forth in the license agreement provided with the software, except for provisions which are contrary to applicable mandatory federal laws.

TRADEMARKS: The trademarks, logos and service marks ("Marks") used herein are the property of Mentor Graphics Corporation or other parties. No one is permitted to use these Marks without the prior written consent of Mentor Graphics or the owner of the Mark, as applicable. The use herein of a third-party Mark is not an attempt to indicate Mentor Graphics as a source of a product, but is intended to indicate a product from, or associated with, a particular third party. A current list of Mentor Graphics’ trademarks may be viewed at: mentor.com/trademarks.

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.

Mentor Graphics Corporation
8005 S.W. Boeckman Road, Wilsonville, Oregon 97070-7777
Telephone: 503.685.7000
Toll-Free Telephone: 800.592.2210
Website: mentor.com
Support Center: support.mentor.com

Send Feedback on Documentation: support.mentor.com/doc_feedback_form
# Table of Contents

## Chapter 1
### Automation and Scripting
- Automating Tasks through Scripting .................................................. 19
- Before You Begin ................................................................................. 20
- Compatible Scripting Languages ......................................................... 23
- Running Scripts .................................................................................. 24
  - Search Order for Scripts .................................................................. 24
  - The scripts.ini File .......................................................................... 25
- Debugging Scripts .............................................................................. 27
  - Invalid Objects in Scripts ............................................................... 27
- Specifying Script and Form Execution .................................................. 29
  - Execution at Application Startup .................................................... 29
  - Executing a Script or Form at Project 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 Xpedition Designer Automation
- Script Examples in this Document ....................................................... 35
  - Example 1: Create/Add Menus ......................................................... 35
  - Example 2: Opening a Data Sheet .................................................. 38
  - Example 3: Opening a Data Sheet .................................................. 40
  - Example 4: Using Objects in Scripts .............................................. 46

## Chapter 3
### Schematic Editor Data Objects
- AddinInfo Object .............................................................................. 53
  - InitiallyDisabled Property (AddinInfo Object) ................................. 56
  - InitiallyVisible Property (AddinInfo Object) ..................................... 57
  - LicenseFeature Property (AddinInfo Object) ..................................... 58
  - Name Property (AddinInfo Object) .................................................. 59
  - Placement Property (AddinInfo Object) ........................................... 60
  - ProgId Property (AddinInfo Object) ................................................ 61
  - RuntimeCreateDecision Property (AddinInfo Object) ..................... 62
  - ShortCutKey Property (AddinInfo Object) ....................................... 63
  - ToolbarButton Property (AddinInfo Object) .................................... 64
- Application Object ............................................................................. 65
  - Activate Method (Application Object) ............................................. 70
<table>
<thead>
<tr>
<th>Method/Property (Application Object)</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddAddin Method</td>
<td>71</td>
</tr>
<tr>
<td>AppendOutput Method</td>
<td>72</td>
</tr>
<tr>
<td>BroadcastDBConfigModified Method</td>
<td>74</td>
</tr>
<tr>
<td>CloseProject Method</td>
<td>75</td>
</tr>
<tr>
<td>CommandsManager Method</td>
<td>76</td>
</tr>
<tr>
<td>DesignComponents Method</td>
<td>77</td>
</tr>
<tr>
<td>DesignNets Method</td>
<td>78</td>
</tr>
<tr>
<td>DesignPaths Method</td>
<td>79</td>
</tr>
<tr>
<td>DesignSearcher Method</td>
<td>80</td>
</tr>
<tr>
<td>GetActiveDesign Method</td>
<td>81</td>
</tr>
<tr>
<td>GetDefaultColor Method</td>
<td>82</td>
</tr>
<tr>
<td>GetProjectData Method</td>
<td>83</td>
</tr>
<tr>
<td>Initialize Method</td>
<td>84</td>
</tr>
<tr>
<td>NewProject Method</td>
<td>85</td>
</tr>
<tr>
<td>OpenBlocks Method</td>
<td>86</td>
</tr>
<tr>
<td>OpenProject Method</td>
<td>87</td>
</tr>
<tr>
<td>OpenURL Method</td>
<td>88</td>
</tr>
<tr>
<td>ParamGetMode Method</td>
<td>89</td>
</tr>
<tr>
<td>ParamGetValue Method</td>
<td>90</td>
</tr>
<tr>
<td>ParamSetMode Method</td>
<td>91</td>
</tr>
<tr>
<td>ParamSetValue Method</td>
<td>92</td>
</tr>
<tr>
<td>PrintProject Method</td>
<td>93</td>
</tr>
<tr>
<td>PushPath Method</td>
<td>95</td>
</tr>
<tr>
<td>Query Method</td>
<td>97</td>
</tr>
<tr>
<td>QueryPages Method</td>
<td>98</td>
</tr>
<tr>
<td>Quit Method</td>
<td>99</td>
</tr>
<tr>
<td>RunISE Method</td>
<td>100</td>
</tr>
<tr>
<td>SchematicSheetDocuments Method</td>
<td>101</td>
</tr>
<tr>
<td>SelectPath Method</td>
<td>102</td>
</tr>
<tr>
<td>SelectPathCompPin Method</td>
<td>104</td>
</tr>
<tr>
<td>SetDefaultColor Method</td>
<td>106</td>
</tr>
<tr>
<td>SetRedraw Method</td>
<td>108</td>
</tr>
<tr>
<td>StartMigration Method</td>
<td>109</td>
</tr>
<tr>
<td>ActiveDocument Property</td>
<td>110</td>
</tr>
<tr>
<td>ActiveView Property</td>
<td>111</td>
</tr>
<tr>
<td>Addins Property</td>
<td>112</td>
</tr>
<tr>
<td>CommandBars Property</td>
<td>113</td>
</tr>
<tr>
<td>CommandLineArguments Property</td>
<td>115</td>
</tr>
<tr>
<td>Interactive Property</td>
<td>116</td>
</tr>
<tr>
<td>QueueSelectEvents Property</td>
<td>117</td>
</tr>
<tr>
<td>ShellCmd Property</td>
<td>118</td>
</tr>
<tr>
<td>SilentMode Property</td>
<td>119</td>
</tr>
<tr>
<td>SourceDocuments Property</td>
<td>120</td>
</tr>
<tr>
<td>StatusBarText Property</td>
<td>121</td>
</tr>
<tr>
<td>Version Property</td>
<td>122</td>
</tr>
<tr>
<td>Visible Property</td>
<td>123</td>
</tr>
<tr>
<td>ActivateView Event</td>
<td>124</td>
</tr>
<tr>
<td>ActivateView2 Event</td>
<td>125</td>
</tr>
<tr>
<td>AfterDocumentOpened Event</td>
<td>126</td>
</tr>
</tbody>
</table>
### Table of Contents

<table>
<thead>
<tr>
<th>Event/Method Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AfterPrintProject Event (Application Object)</td>
<td>127</td>
</tr>
<tr>
<td>AfterSheetRead Event (Application Object)</td>
<td>128</td>
</tr>
<tr>
<td>AfterSheetReRead Event (Application Object)</td>
<td>129</td>
</tr>
<tr>
<td>BeforeDocumentOpened Event (Application Object)</td>
<td>130</td>
</tr>
<tr>
<td>BeforePrintProject Event (Application Object)</td>
<td>131</td>
</tr>
<tr>
<td>BeforeProjectChanged Event (Application Object)</td>
<td>132</td>
</tr>
<tr>
<td>BlockLocked Event (Application Object)</td>
<td>133</td>
</tr>
<tr>
<td>BlockModified Event (Application Object)</td>
<td>134</td>
</tr>
<tr>
<td>CreateObject Event (Application Object)</td>
<td>135</td>
</tr>
<tr>
<td>DeactivateView Event (Application Object)</td>
<td>136</td>
</tr>
<tr>
<td>DeactivateView2 Event (Application Object)</td>
<td>137</td>
</tr>
<tr>
<td>Delete Event (Application Object)</td>
<td>138</td>
</tr>
<tr>
<td>DocumentClose Event (Application Object)</td>
<td>139</td>
</tr>
<tr>
<td>LockRequest Event (Application Object)</td>
<td>140</td>
</tr>
<tr>
<td>MouseMoved Event (Application Object)</td>
<td>141</td>
</tr>
<tr>
<td>PaintRegion Event (Application Object)</td>
<td>142</td>
</tr>
<tr>
<td>PrintFile Event (Application Object)</td>
<td>143</td>
</tr>
<tr>
<td>ProjectChanged Event (Application Object)</td>
<td>144</td>
</tr>
<tr>
<td>ProjectClosed Event (Application Object)</td>
<td>145</td>
</tr>
<tr>
<td>Select Event (Application Object)</td>
<td>146</td>
</tr>
<tr>
<td>Shutdown Event (Application Object)</td>
<td>147</td>
</tr>
<tr>
<td>SourceDocumentSave Event (Application Object)</td>
<td>148</td>
</tr>
<tr>
<td>SourceFileModified Event (Application Object)</td>
<td>149</td>
</tr>
<tr>
<td>Startup Event (Application Object)</td>
<td>150</td>
</tr>
<tr>
<td>SymbolPreviewed Event (Application Object)</td>
<td>151</td>
</tr>
<tr>
<td>Unlock Event (Application Object)</td>
<td>152</td>
</tr>
<tr>
<td>Arc Object</td>
<td>153</td>
</tr>
<tr>
<td>GetLocation Method (Arc Object)</td>
<td>154</td>
</tr>
<tr>
<td>GetObjectColor Method (Arc Object)</td>
<td>155</td>
</tr>
<tr>
<td>IsColorAutomatic Method (Arc Object)</td>
<td>156</td>
</tr>
<tr>
<td>SetAutomaticColor Method (Arc Object)</td>
<td>157</td>
</tr>
<tr>
<td>SetLocation Method (Arc Object)</td>
<td>158</td>
</tr>
<tr>
<td>SetObjectColor Method (Arc Object)</td>
<td>159</td>
</tr>
<tr>
<td>Application Property (Arc Object)</td>
<td>160</td>
</tr>
<tr>
<td>LineStyle Property (Arc Object)</td>
<td>161</td>
</tr>
<tr>
<td>Parent Property (Arc Object)</td>
<td>162</td>
</tr>
<tr>
<td>Selected Property (Arc Object)</td>
<td>163</td>
</tr>
<tr>
<td>Type Property (Arc Object)</td>
<td>164</td>
</tr>
<tr>
<td>Attribute Object</td>
<td>165</td>
</tr>
<tr>
<td>Delete Method (Attribute Object)</td>
<td>167</td>
</tr>
<tr>
<td>DeleteInstanceValue Method (Attribute Object)</td>
<td>168</td>
</tr>
<tr>
<td>GetLocation Method (Attribute Object)</td>
<td>169</td>
</tr>
<tr>
<td>GetOatFull Method (Attribute Object)</td>
<td>170</td>
</tr>
<tr>
<td>GetObjectColor Method (Attribute Object)</td>
<td>171</td>
</tr>
<tr>
<td>IsColorAutomatic Method (Attribute Object)</td>
<td>172</td>
</tr>
<tr>
<td>SetAutomaticColor Method (Attribute Object)</td>
<td>173</td>
</tr>
<tr>
<td>SetLocation Method (Attribute Object)</td>
<td>174</td>
</tr>
<tr>
<td>SetObjectColor Method (Attribute Object)</td>
<td>175</td>
</tr>
<tr>
<td>Application Property (Attribute Object)</td>
<td>176</td>
</tr>
<tr>
<td>Method/Property</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Block Object</td>
<td>191</td>
</tr>
<tr>
<td>AddArc Method (Block Object)</td>
<td>194</td>
</tr>
<tr>
<td>AddAttribute Method (Block Object)</td>
<td>195</td>
</tr>
<tr>
<td>AddBatchAttributes Method (Block Object)</td>
<td>196</td>
</tr>
<tr>
<td>AddBox Method (Block Object)</td>
<td>197</td>
</tr>
<tr>
<td>AddCircle Method (Block Object)</td>
<td>198</td>
</tr>
<tr>
<td>AddFub Method (Block Object)</td>
<td>199</td>
</tr>
<tr>
<td>AddLine Method (Block Object)</td>
<td>200</td>
</tr>
<tr>
<td>AddLine2 Method (Block Object)</td>
<td>201</td>
</tr>
<tr>
<td>AddNet Method (Block Object)</td>
<td>202</td>
</tr>
<tr>
<td>AddNetEx Method (Block Object)</td>
<td>204</td>
</tr>
<tr>
<td>AddPartInstance Method (Block Object)</td>
<td>206</td>
</tr>
<tr>
<td>AddPin Method (Block Object)</td>
<td>207</td>
</tr>
<tr>
<td>AddPinAtLocation Method (Block Object)</td>
<td>208</td>
</tr>
<tr>
<td>AddSymbolInstance Method (Block Object)</td>
<td>209</td>
</tr>
<tr>
<td>AddText Method (Block Object)</td>
<td>210</td>
</tr>
<tr>
<td>ApplySymbolUpdate Method (Block Object)</td>
<td>211</td>
</tr>
<tr>
<td>ChangeBorder Method (Block Object)</td>
<td>212</td>
</tr>
<tr>
<td>ChangeComponent Method (Block Object)</td>
<td>213</td>
</tr>
<tr>
<td>ChangeComponentPreserveRefdes Method (Block Object)</td>
<td>214</td>
</tr>
<tr>
<td>ClearHighlight Method (Block Object)</td>
<td>215</td>
</tr>
<tr>
<td>DeleteBorder Method (Block Object)</td>
<td>216</td>
</tr>
<tr>
<td>DeleteSelected Method (Block Object)</td>
<td>217</td>
</tr>
<tr>
<td>DeSelectAll Method (Block Object)</td>
<td>218</td>
</tr>
<tr>
<td>FindAttribute Method (Block Object)</td>
<td>219</td>
</tr>
<tr>
<td>GetBatchAttributes Method (Block Object)</td>
<td>220</td>
</tr>
<tr>
<td>GetBboxPoint Method (Block Object)</td>
<td>221</td>
</tr>
<tr>
<td>GetChildBlock Method (Block Object)</td>
<td>222</td>
</tr>
<tr>
<td>GetName Method (Block Object)</td>
<td>223</td>
</tr>
<tr>
<td>InsertBorder Method (Block Object)</td>
<td>224</td>
</tr>
<tr>
<td>PromoteSymbolNumbers Method (Block Object)</td>
<td>225</td>
</tr>
<tr>
<td>RepositionAttributesAsOnSymbol Method (Block Object)</td>
<td>226</td>
</tr>
<tr>
<td>SetZSheetSize Method (Block Object)</td>
<td>227</td>
</tr>
<tr>
<td>UpdateBorder Method (Block Object)</td>
<td>228</td>
</tr>
<tr>
<td>Application Property (Block Object)</td>
<td>229</td>
</tr>
</tbody>
</table>
## Table of Contents

- **Attributes Property (Block Object)** ................................................................. 230
- **DataType Property (Block Object)** ................................................................. 231
- **IsFub Property (Block Object)** ........................................................................... 232
- **LibraryName Property (Block Object)** ............................................................. 233
- **OpenMode Property (Block Object)** ............................................................... 234
- **Parent Property (Block Object)** ......................................................................... 235
- **SheetSize Property (Block Object)** ................................................................. 237
- **SymbolType Property (Block Object)** ............................................................. 238
- **Type Property (Block Object)** ......................................................................... 239
- **UpdateOutline Method (Block Object)** ......................................................... 240
- **Box Object** ........................................................................................................ 241
- **GetLocation Method (Box Object)** ................................................................. 242
- **GetObjectColor Method (Box Object)** ......................................................... 243
- **IsColorAutomatic Method (Box Object)** ..................................................... 244
- **SetAutomaticColor Method (Box Object)** .................................................... 245
- **SetLocation Method (Box Object)** ................................................................. 246
- **SetObjectColor Method (Box Object)** .......................................................... 247
- **Application Property (Box Object)** ............................................................. 248
- **FillStyle Property (Box Object)** ................................................................. 249
- **LineStyle Property (Box Object)** ................................................................. 250
- **Parent Property (Box Object)** ......................................................................... 251
- **Selected Property (Box Object)** ...................................................................... 252
- **Type Property (Box Object)** ......................................................................... 253
- **CColor Object** .................................................................................................. 254
  - **b Property (CColor Object)** ......................................................................... 255
  - **g Property (CColor Object)** ......................................................................... 256
  - **r Property (CColor Object)** ......................................................................... 257
- **Circle Object** .................................................................................................... 258
  - **GetCenter Method (Circle Object)** ............................................................. 259
  - **GetObjectColor Method (Circle Object)** ............................................... 260
  - **IsColorAutomatic Method (Circle Object)** .................................................. 261
  - **SetAutomaticColor Method (Circle Object)** .................................................. 262
  - **SetCenter Method (Circle Object)** ............................................................ 263
  - **SetObjectColor Method (Circle Object)** ..................................................... 264
  - **Application Property (Circle Object)** .......................................................... 265
  - **FillStyle Property (Circle Object)** ............................................................. 266
  - **LineStyle Property (Circle Object)** ............................................................ 267
  - **Parent Property (Circle Object)** ................................................................. 268
  - **Radius Property (Circle Object)** ................................................................. 269
  - **Selected Property (Circle Object)** ............................................................. 270
  - **Type Property (Circle Object)** ................................................................ 271
- **CommandsManager Object** ................................................................................. 272
  - **CommandDisable Method (CommandsManager Object)** .......................... 273
  - **CommandEnable Method (CommandsManager Object)** ............................ 274
  - **CommandRemove Method (CommandsManager Object)** ............................ 275
  - **ExecuteCommand Method (CommandsManager Object)** .......................... 276
  - **ExecuteMenuCommand Method (CommandsManager Object)** ................. 277
  - **RegisterOLECommand Method (CommandsManager Object)** .................. 278
<table>
<thead>
<tr>
<th>Method/Property (Object Name)</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>UnregisterOLECommand Method (CommandsManager Object)</td>
<td>280</td>
</tr>
<tr>
<td>Component Object</td>
<td>279</td>
</tr>
<tr>
<td>AddAttribute Method (Component Object)</td>
<td>283</td>
</tr>
<tr>
<td>AddBatchAttributes Method (Component Object)</td>
<td>284</td>
</tr>
<tr>
<td>AddBatchOats Method (Component Object)</td>
<td>285</td>
</tr>
<tr>
<td>AddLabel Method (Component Object)</td>
<td>286</td>
</tr>
<tr>
<td>AddOat Method (Component Object)</td>
<td>287</td>
</tr>
<tr>
<td>FindAttribute Method (Component Object)</td>
<td>288</td>
</tr>
<tr>
<td>GetBatchAttributes Method (Component Object)</td>
<td>289</td>
</tr>
<tr>
<td>GetBatchOats Method (Component Object)</td>
<td>290</td>
</tr>
<tr>
<td>GetBoxPoint Method (Component Object)</td>
<td>291</td>
</tr>
<tr>
<td>GetConnections Method (Component Object)</td>
<td>292</td>
</tr>
<tr>
<td>GetLocation Method (Component Object)</td>
<td>293</td>
</tr>
<tr>
<td>GetName Method (Component Object)</td>
<td>294</td>
</tr>
<tr>
<td>SetLocation Method (Component Object)</td>
<td>295</td>
</tr>
<tr>
<td>Application Property (Component Object)</td>
<td>296</td>
</tr>
<tr>
<td>Attributes Property (Component Object)</td>
<td>297</td>
</tr>
<tr>
<td>Id Property (Component Object)</td>
<td>298</td>
</tr>
<tr>
<td>Label Property (Component Object)</td>
<td>299</td>
</tr>
<tr>
<td>Orientation Property (Component Object)</td>
<td>300</td>
</tr>
<tr>
<td>Parent Property (Component Object)</td>
<td>301</td>
</tr>
<tr>
<td>Refdes Property (Component Object)</td>
<td>302</td>
</tr>
<tr>
<td>Scale Property (Component Object)</td>
<td>303</td>
</tr>
<tr>
<td>Selected Property (Component Object)</td>
<td>304</td>
</tr>
<tr>
<td>SymbolBlock Property (Component Object)</td>
<td>305</td>
</tr>
<tr>
<td>Type Property (Component Object)</td>
<td>306</td>
</tr>
<tr>
<td>UID Property (Component Object)</td>
<td>307</td>
</tr>
<tr>
<td>ComponentPin Object</td>
<td>308</td>
</tr>
<tr>
<td>AddAttribute Method (ComponentPin Object)</td>
<td>309</td>
</tr>
<tr>
<td>AddOAT Method (ComponentPin Object)</td>
<td>310</td>
</tr>
<tr>
<td>FindAttribute Method (ComponentPin Object)</td>
<td>311</td>
</tr>
<tr>
<td>GetLocation Method (ComponentPin Object)</td>
<td>312</td>
</tr>
<tr>
<td>Application Property (ComponentPin Object)</td>
<td>313</td>
</tr>
<tr>
<td>Attributes Property (ComponentPin Object)</td>
<td>314</td>
</tr>
<tr>
<td>Component Property (ComponentPin Object)</td>
<td>315</td>
</tr>
<tr>
<td>Connection Property (ComponentPin Object)</td>
<td>316</td>
</tr>
<tr>
<td>Number Property (ComponentPin Object)</td>
<td>317</td>
</tr>
<tr>
<td>Parent Property (ComponentPin Object)</td>
<td>318</td>
</tr>
<tr>
<td>Pin Property (ComponentPin Object)</td>
<td>319</td>
</tr>
<tr>
<td>Selected Property (ComponentPin Object)</td>
<td>320</td>
</tr>
<tr>
<td>Side Property (ComponentPin Object)</td>
<td>321</td>
</tr>
<tr>
<td>Type Property (ComponentPin Object)</td>
<td>322</td>
</tr>
<tr>
<td>Connection Object</td>
<td>323</td>
</tr>
<tr>
<td>CompPin Property (Connection Object)</td>
<td>324</td>
</tr>
<tr>
<td>Net Property (Connection Object)</td>
<td>325</td>
</tr>
<tr>
<td>Ripper Property (Connection Object)</td>
<td>326</td>
</tr>
<tr>
<td>Segment Property (Connection Object)</td>
<td>327</td>
</tr>
<tr>
<td>HDLSourceDocument Object</td>
<td>328</td>
</tr>
<tr>
<td>BookmarkLine Method (HDLSourceDocument Object)</td>
<td>329</td>
</tr>
<tr>
<td>Table of Contents</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td></td>
</tr>
<tr>
<td>GotoLine Method (HDLSourceDocument Object)</td>
<td>330</td>
</tr>
<tr>
<td>Name Property (HDLSourceDocument Object)</td>
<td>331</td>
</tr>
<tr>
<td>Path Property (HDLSourceDocument Object)</td>
<td>332</td>
</tr>
<tr>
<td>Label Object</td>
<td>333</td>
</tr>
<tr>
<td>GetLocation Method (Label Object)</td>
<td>335</td>
</tr>
<tr>
<td>GetObjectColor Method (Label Object)</td>
<td>336</td>
</tr>
<tr>
<td>IsColorAutomatic Method (Label Object)</td>
<td>337</td>
</tr>
<tr>
<td>SetAutomaticColor Method (Label Object)</td>
<td>338</td>
</tr>
<tr>
<td>SetLocation Method (Label Object)</td>
<td>339</td>
</tr>
<tr>
<td>SetObjectColor Method (Label Object)</td>
<td>340</td>
</tr>
<tr>
<td>Application Property (Label Object)</td>
<td>341</td>
</tr>
<tr>
<td>Font Property (Label Object)</td>
<td>342</td>
</tr>
<tr>
<td>Orientation Property (Label Object)</td>
<td>343</td>
</tr>
<tr>
<td>Origin Property (Label Object)</td>
<td>344</td>
</tr>
<tr>
<td>Parent Property (Label Object)</td>
<td>345</td>
</tr>
<tr>
<td>ResolvedName Property (Label Object)</td>
<td>346</td>
</tr>
<tr>
<td>Scope Property (Label Object)</td>
<td>347</td>
</tr>
<tr>
<td>Selected Property (Label Object)</td>
<td>348</td>
</tr>
<tr>
<td>Sense Property (Label Object)</td>
<td>349</td>
</tr>
<tr>
<td>Size Property (Label Object)</td>
<td>350</td>
</tr>
<tr>
<td>TextString Property (Label Object)</td>
<td>351</td>
</tr>
<tr>
<td>Type Property (Label Object)</td>
<td>352</td>
</tr>
<tr>
<td>Visible Property (Label Object)</td>
<td>353</td>
</tr>
<tr>
<td>Line Object</td>
<td>354</td>
</tr>
<tr>
<td>AddPoint Method (Line Object)</td>
<td>355</td>
</tr>
<tr>
<td>GetNumPoints Method (Line Object)</td>
<td>356</td>
</tr>
<tr>
<td>GetObjectColor Method (Line Object)</td>
<td>357</td>
</tr>
<tr>
<td>GetPoint Method (Line Object)</td>
<td>358</td>
</tr>
<tr>
<td>IsColorAutomatic Method (Line Object)</td>
<td>359</td>
</tr>
<tr>
<td>SetAutomaticColor Method (Line Object)</td>
<td>360</td>
</tr>
<tr>
<td>SetObjectColor Method (Line Object)</td>
<td>361</td>
</tr>
<tr>
<td>Application Property (Line Object)</td>
<td>362</td>
</tr>
<tr>
<td>LineStyle Property (Line Object)</td>
<td>363</td>
</tr>
<tr>
<td>Parent Property (Line Object)</td>
<td>364</td>
</tr>
<tr>
<td>Selected Property (Line Object)</td>
<td>365</td>
</tr>
<tr>
<td>Type Property (Line Object)</td>
<td>366</td>
</tr>
<tr>
<td>Net Object</td>
<td>367</td>
</tr>
<tr>
<td>AddAttribute Method (Net Object)</td>
<td>369</td>
</tr>
<tr>
<td>AddLabel Method (Net Object)</td>
<td>370</td>
</tr>
<tr>
<td>Connections Method (Net Object)</td>
<td>371</td>
</tr>
<tr>
<td>FindAttribute Method (Net Object)</td>
<td>372</td>
</tr>
<tr>
<td>GetConnectedLabel Method (Net Object)</td>
<td>373</td>
</tr>
<tr>
<td>GetConnectedNetName Method (Net Object)</td>
<td>374</td>
</tr>
<tr>
<td>GetLabel Method (Net Object)</td>
<td>375</td>
</tr>
<tr>
<td>GetObjectColor Method (Net Object)</td>
<td>376</td>
</tr>
<tr>
<td>GetRippers Method (Net Object)</td>
<td>377</td>
</tr>
<tr>
<td>GetSegments Method (Net Object)</td>
<td>378</td>
</tr>
<tr>
<td>GetSignals Method (Net Object)</td>
<td>379</td>
</tr>
<tr>
<td>GetSingleJointLocs Method (Net Object)</td>
<td>380</td>
</tr>
<tr>
<td>Method/Property</td>
<td>Page</td>
</tr>
<tr>
<td>----------------</td>
<td>------</td>
</tr>
<tr>
<td>IsColorAutomatic Method (Net Object)</td>
<td>381</td>
</tr>
<tr>
<td>IsSegmentSelected Method (Net Object)</td>
<td>382</td>
</tr>
<tr>
<td>SelectSegment Method (Net Object)</td>
<td>383</td>
</tr>
<tr>
<td>SelectSegmentByJointLoc Method (Net Object)</td>
<td>384</td>
</tr>
<tr>
<td>SetAutomaticColor Method (Net Object)</td>
<td>385</td>
</tr>
<tr>
<td>SetObjectColor Method (Net Object)</td>
<td>386</td>
</tr>
<tr>
<td>Application Property (Net Object)</td>
<td>387</td>
</tr>
<tr>
<td>Attributes Property (Net Object)</td>
<td>388</td>
</tr>
<tr>
<td>Id Property (Net Object)</td>
<td>389</td>
</tr>
<tr>
<td>LineStyle Property (Net Object)</td>
<td>390</td>
</tr>
<tr>
<td>Parent Property (Net Object)</td>
<td>391</td>
</tr>
<tr>
<td>Selected Property (Net Object)</td>
<td>392</td>
</tr>
<tr>
<td>Type Property (Net Object)</td>
<td>393</td>
</tr>
<tr>
<td>UID Property (Net Object)</td>
<td>394</td>
</tr>
<tr>
<td>AddPDBPartition Method (PDBPartitions Object)</td>
<td>395</td>
</tr>
<tr>
<td>GetPDBPartition Method (PDBPartitions Object)</td>
<td>396</td>
</tr>
<tr>
<td>GetPDBPartitionsArray Method (PDBPartitions Object)</td>
<td>397</td>
</tr>
<tr>
<td>InsertPDBPartition Method (PDBPartitions Object)</td>
<td>398</td>
</tr>
<tr>
<td>PDBPartitionExists Method (PDBPartitions Object)</td>
<td>399</td>
</tr>
<tr>
<td>RemovePDBPartitionByIndex Method (PDBPartitions Object)</td>
<td>400</td>
</tr>
<tr>
<td>RemovePDBPartitionByName Method (PDBPartitions Object)</td>
<td>401</td>
</tr>
<tr>
<td>Pin Object</td>
<td>402</td>
</tr>
<tr>
<td>AddAttribute Method (Pin Object)</td>
<td>403</td>
</tr>
<tr>
<td>FindAttribute Method (Pin Object)</td>
<td>404</td>
</tr>
<tr>
<td>GetLocation Method (Pin Object)</td>
<td>405</td>
</tr>
<tr>
<td>GetName Method (Pin Object)</td>
<td>406</td>
</tr>
<tr>
<td>GetObjectColor Method (Pin Object)</td>
<td>407</td>
</tr>
<tr>
<td>GetObjectColor Method (Pin Object)</td>
<td>408</td>
</tr>
<tr>
<td>SetLocation Method (Pin Object)</td>
<td>409</td>
</tr>
<tr>
<td>Application Property (Pin Object)</td>
<td>410</td>
</tr>
<tr>
<td>Attributes Property (Pin Object)</td>
<td>411</td>
</tr>
<tr>
<td>Id Property (Pin Object)</td>
<td>412</td>
</tr>
<tr>
<td>Label Property (Pin Object)</td>
<td>413</td>
</tr>
<tr>
<td>Parent Property (Pin Object)</td>
<td>414</td>
</tr>
<tr>
<td>Selected Property (Pin Object)</td>
<td>415</td>
</tr>
<tr>
<td>Sense Property (Pin Object)</td>
<td>416</td>
</tr>
<tr>
<td>Side Property (Pin Object)</td>
<td>417</td>
</tr>
<tr>
<td>Type Property (Pin Object)</td>
<td>418</td>
</tr>
<tr>
<td>UID Property (Pin Object)</td>
<td>419</td>
</tr>
<tr>
<td>Point Object</td>
<td>420</td>
</tr>
<tr>
<td>X Property (Point Object)</td>
<td>421</td>
</tr>
<tr>
<td>Y Property (Point Object)</td>
<td>422</td>
</tr>
<tr>
<td>ProjectData Object</td>
<td>423</td>
</tr>
<tr>
<td>AddICDBDesign Method (ProjectData Object)</td>
<td>424</td>
</tr>
<tr>
<td>GetBordersFilePath Method (ProjectData Object)</td>
<td>425</td>
</tr>
<tr>
<td>GetBusContentsFilePath Method (ProjectData Object)</td>
<td>426</td>
</tr>
<tr>
<td>GetICDBDesignRootBlock Method (ProjectData Object)</td>
<td>427</td>
</tr>
<tr>
<td>GetICDBDesigns Method (ProjectData Object)</td>
<td>428</td>
</tr>
<tr>
<td>GetICDBDesignType Method (ProjectData Object)</td>
<td>429</td>
</tr>
<tr>
<td>Method/Property (Object)</td>
<td>Page</td>
</tr>
<tr>
<td>---------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>GetiCDBDiscardFilePath Method (ProjectData Object)</td>
<td>432</td>
</tr>
<tr>
<td>GetPCBDesignPath Method (ProjectData Object)</td>
<td>433</td>
</tr>
<tr>
<td>GetPDBPartitions Method (ProjectData Object)</td>
<td>434</td>
</tr>
<tr>
<td>GetPinComponentsFilePath Method (ProjectData Object)</td>
<td>435</td>
</tr>
<tr>
<td>GetProjectFilePath Method (ProjectData Object)</td>
<td>436</td>
</tr>
<tr>
<td>GetProjectPath Method (ProjectData Object)</td>
<td>437</td>
</tr>
<tr>
<td>GetSearchPathScheme Method (ProjectData Object)</td>
<td>439</td>
</tr>
<tr>
<td>GetSymbolPartitions Method (ProjectData Object)</td>
<td>440</td>
</tr>
<tr>
<td>RemoveiCDBDesign Method (ProjectData Object)</td>
<td>441</td>
</tr>
<tr>
<td>RenameiCDBDesign Method (ProjectData Object)</td>
<td>442</td>
</tr>
<tr>
<td>SetBordersFilePath Method (ProjectData Object)</td>
<td>443</td>
</tr>
<tr>
<td>SetBusContentsFilePath Method (ProjectData Object)</td>
<td>444</td>
</tr>
<tr>
<td>SetiCDBDesignRootBlock Method (ProjectData Object)</td>
<td>445</td>
</tr>
<tr>
<td>SetiCDBDesignType Method (ProjectData Object)</td>
<td>446</td>
</tr>
<tr>
<td>SetiCDBDiscardFilePath Method (ProjectData Object)</td>
<td>447</td>
</tr>
<tr>
<td>SetPCBDesignPath Method (ProjectData Object)</td>
<td>448</td>
</tr>
<tr>
<td>SetPinComponentsFilePath Method (ProjectData Object)</td>
<td>449</td>
</tr>
<tr>
<td>SetSearchPathScheme Method (ProjectData Object)</td>
<td>450</td>
</tr>
<tr>
<td>UpdateOtherObjects Method (ProjectData Object)</td>
<td>451</td>
</tr>
<tr>
<td>CentralLibraryPath Property (ProjectData Object)</td>
<td>452</td>
</tr>
<tr>
<td>iCDBDir Property (ProjectData Object)</td>
<td>453</td>
</tr>
<tr>
<td>Rect Object</td>
<td>454</td>
</tr>
<tr>
<td>Top Property (Rect Object)</td>
<td>455</td>
</tr>
<tr>
<td>Right Property (Rect Object)</td>
<td>456</td>
</tr>
<tr>
<td>Bottom Property (Rect Object)</td>
<td>457</td>
</tr>
<tr>
<td>Left Property (Rect Object)</td>
<td>458</td>
</tr>
<tr>
<td>Ripper Object</td>
<td>459</td>
</tr>
<tr>
<td>GetConnectedObject Method (Ripper Object)</td>
<td>460</td>
</tr>
<tr>
<td>GetConnectedObjects Method (Ripper Object)</td>
<td>461</td>
</tr>
<tr>
<td>GetMappedSignal Method (Ripper Object)</td>
<td>462</td>
</tr>
<tr>
<td>SchematicSheetDocument Object</td>
<td>463</td>
</tr>
<tr>
<td>Activate Method (SchematicSheetDocument Object)</td>
<td>465</td>
</tr>
<tr>
<td>Close Method (SchematicSheetDocument Object)</td>
<td>466</td>
</tr>
<tr>
<td>DiscardSymbolChanges Method (SchematicSheetDocument Object)</td>
<td>467</td>
</tr>
<tr>
<td>ExportMetafile Method (SchematicSheetDocument Object)</td>
<td>468</td>
</tr>
<tr>
<td>GetViews Method (SchematicSheetDocument Object)</td>
<td>469</td>
</tr>
<tr>
<td>IsReadOnly Method (SchematicSheetDocument Object)</td>
<td>470</td>
</tr>
<tr>
<td>Print Method (SchematicSheetDocument Object)</td>
<td>471</td>
</tr>
<tr>
<td>ReRead Method (SchematicSheetDocument Object)</td>
<td>472</td>
</tr>
<tr>
<td>Save Method (SchematicSheetDocument Object)</td>
<td>473</td>
</tr>
<tr>
<td>SaveAs Method (SchematicSheetDocument Object)</td>
<td>474</td>
</tr>
<tr>
<td>UpdateSymbolInDesign Method (SchematicSheetDocument Object)</td>
<td>475</td>
</tr>
<tr>
<td>Application Property (SchematicSheetDocument Object)</td>
<td>476</td>
</tr>
<tr>
<td>FullName Property (SchematicSheetDocument Object)</td>
<td>477</td>
</tr>
<tr>
<td>Name Property (SchematicSheetDocument Object)</td>
<td>478</td>
</tr>
<tr>
<td>Parent Property (SchematicSheetDocument Object)</td>
<td>479</td>
</tr>
<tr>
<td>Segment Object</td>
<td>480</td>
</tr>
<tr>
<td>GetJointType Method (Segment Object)</td>
<td>481</td>
</tr>
<tr>
<td>Class</td>
<td>Method</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Text Object</td>
<td>GetLocation Method (Text Object)</td>
</tr>
<tr>
<td></td>
<td>GetObjectColor Method (Text Object)</td>
</tr>
<tr>
<td></td>
<td>IsColorAutomatic Method (Text Object)</td>
</tr>
<tr>
<td></td>
<td>SetAutomaticColor Method (Text Object)</td>
</tr>
<tr>
<td></td>
<td>SetLocation Method (Text Object)</td>
</tr>
<tr>
<td></td>
<td>SetObjectColor Method (Text Object)</td>
</tr>
<tr>
<td></td>
<td>Application Property (Text Object)</td>
</tr>
<tr>
<td></td>
<td>Font Property (Text Object)</td>
</tr>
<tr>
<td></td>
<td>Orientation Property (Text Object)</td>
</tr>
<tr>
<td></td>
<td>Origin Property (Text Object)</td>
</tr>
<tr>
<td></td>
<td>Parent Property (Text Object)</td>
</tr>
<tr>
<td></td>
<td>Selected Property (Text Object)</td>
</tr>
<tr>
<td></td>
<td>Size Property (Text Object)</td>
</tr>
<tr>
<td></td>
<td>TextString Property (Text Object)</td>
</tr>
<tr>
<td></td>
<td>Type Property (Text Object)</td>
</tr>
<tr>
<td>View Object</td>
<td>Activate Method (View Object)</td>
</tr>
<tr>
<td></td>
<td>AddAttributeMoveMode Method (View Object)</td>
</tr>
<tr>
<td></td>
<td>Application Method (View Object)</td>
</tr>
<tr>
<td></td>
<td>BufferCopy Method (View Object)</td>
</tr>
<tr>
<td></td>
<td>BufferCut Method (View Object)</td>
</tr>
<tr>
<td></td>
<td>BufferPaste Method (View Object)</td>
</tr>
<tr>
<td></td>
<td>BufferPasteXY Method (View Object)</td>
</tr>
<tr>
<td></td>
<td>ComputeMBB Method (View Object)</td>
</tr>
<tr>
<td></td>
<td>Document Method (View Object)</td>
</tr>
<tr>
<td></td>
<td>GetJointLocs Method (View Object)</td>
</tr>
<tr>
<td></td>
<td>GetName Method (View Object)</td>
</tr>
<tr>
<td></td>
<td>GetSelectedNetName Method (View Object)</td>
</tr>
<tr>
<td></td>
<td>GetTopLevelDesignName Method (View Object)</td>
</tr>
<tr>
<td></td>
<td>ModifyVisibility Method (View Object)</td>
</tr>
<tr>
<td></td>
<td>Query Method (View Object)</td>
</tr>
<tr>
<td></td>
<td>Refresh Method (View Object)</td>
</tr>
<tr>
<td></td>
<td>SelectbyName Method (View Object)</td>
</tr>
<tr>
<td></td>
<td>SymbolPartitions Object</td>
</tr>
<tr>
<td></td>
<td>AppendSymbolPartition Method (SymbolPartitions Object)</td>
</tr>
<tr>
<td></td>
<td>GetSymbolPartition Method (SymbolPartitions Object)</td>
</tr>
<tr>
<td></td>
<td>GetSymbolPartitionsArray Method (SymbolPartitions Object)</td>
</tr>
<tr>
<td></td>
<td>GetSymbolPartitionsCount Method (SymbolPartitions Object)</td>
</tr>
<tr>
<td></td>
<td>InsertSymbolPartition Method (SymbolPartitions Object)</td>
</tr>
<tr>
<td></td>
<td>RemoveSymbolPartitionByIndex Method (SymbolPartitions Object)</td>
</tr>
<tr>
<td></td>
<td>RemoveSymbolPartitionByName Method (SymbolPartitions Object)</td>
</tr>
<tr>
<td></td>
<td>SymbolPartitionExists Method (SymbolPartitions Object)</td>
</tr>
</tbody>
</table>
Table of Contents

SelectbyName2 Method (View Object) .................................................. 533
SelectObject Method (View Object) .................................................... 534
SelectSegmentByJointLoc Method (View Object) ................................. 536
SelectText Method (View Object) ...................................................... 537
SetCenter Method (View Object) ....................................................... 538
ViewFull Method (View Object) ......................................................... 539
ZoomIn Method (View Object) ........................................................ 540
ZoomOut Method (View Object) ........................................................ 541
ZoomSelect Method (View Object) .................................................... 542
Block Property (View Object) .......................................................... 543
TopBlock Property (View Object) ...................................................... 544
Viewport Property (View Object) ...................................................... 545
OnActivate Event (View Object) ....................................................... 546
OnSelect Event (View Object) .......................................................... 547
Viewport Object ................................................................................. 548
Arc Method (Viewport Object) .......................................................... 550
Arrow Method (Viewport Object) ....................................................... 551
Box Method (Viewport Object) ........................................................ 552
Circle Method (Viewport Object) ....................................................... 553
Ellipse Method (Viewport Object) ..................................................... 554
EraseRectangle Method (Viewport Object) ......................................... 555
GetObjectColor Method (Viewport Object) .......................................... 556
Line Method (Viewport Object) ........................................................ 557
PixelRectangle Method (Viewport Object) .......................................... 558
PixelToUser Method (Viewport Object) .............................................. 559
Point Method (Viewport Object) ........................................................ 560
PolyLine Method (Viewport Object) ................................................... 561
SetClipRectangle Method (Viewport Object) ..................................... 562
SetObjectColor Method (Viewport Object) ......................................... 563
Spline Method (Viewport Object) ...................................................... 564
Text Method (Viewport Object) ........................................................ 566
UserRectangle Method (Viewport Object) ........................................... 567
UserToPixel Method (Viewport Object) .............................................. 568
FillStyle Property (Viewport Object) ................................................. 569
LineCap Property (Viewport Object) ................................................. 570
LineJoin Property (Viewport Object) ................................................. 571
LinePattern Property (Viewport Object) ............................................ 572
LineThickness Property (Viewport Object) ....................................... 573
RasterMode Property (Viewport Object) ............................................ 574
TextAngle Property (Viewport Object) .............................................. 575
TextFont Property (Viewport Object) ............................................... 576
TextSize Property (Viewport Object) ................................................. 577

Chapter 4
Xpedition Designer Schematic Editor Object Collections ........................................ 579
HDLSourceDocuments Collection ....................................................... 580
Item Method (HDLSourceDocuments Collection) ................................ 581
New Method (HDLSourceDocuments Collection) .................................. 582
Table of Contents

Open Method (HDLSourceDocuments Collection) ................................................. 583
Remove Method (HDLSourceDocuments Collection) ............................................. 584
RemoveAll Method (HDLSourceDocuments Collection) ......................................... 585
SaveAll Method (HDLSourceDocuments Collection) ............................................. 586
Count Property (HDLSourceDocuments Collection) ............................................ 587
SchematicSheetDocuments Collection ................................................................. 588
Close Method (SchematicSheetDocuments Collection) .......................................... 590
CopyToClipboard Method (SchematicSheetDocuments Collection) ......................... 591
DeleteSheet Method (SchematicSheetDocuments Collection) ............................... 594
GetAvailableSchematics Method (SchematicSheetDocuments Collection) ............... 595
GetAvailableSheets Method (SchematicSheetDocuments Collection) ..................... 596
InsertSheet Method (SchematicSheetDocuments Collection) .............................. 597
IsSymbolUnderEdit Method (SchematicSheetDocuments Collection) ..................... 598
Open Method (SchematicSheetDocuments Collection) ........................................... 600
Open_Hierarchically Method (SchematicSheetDocuments Collection) ................... 601
OpenSymbol Method (SchematicSheetDocuments Collection) .............................. 602
PasteFromClipboard Method (SchematicSheetDocuments Collection) ................. 603
Application Property (SchematicSheetDocuments Collection) ............................ 604
Count Property (SchematicSheetDocuments Collection) ..................................... 605
Parent Property (SchematicSheetDocuments Collection) ..................................... 606
StringCollection Collection .............................................................................. 607
Item Method (StringCollection Collection) ......................................................... 608
Remove Method (StringCollection Collection) .................................................... 609
Count Property (StringCollection Collection) ...................................................... 610
StringList Collection ......................................................................................... 611
Append Method (StringList Collection) .............................................................. 612
Clear Method (StringList Collection) ................................................................. 613
GetCount Method (StringList Collection) ........................................................... 614
GetItem Method (StringList Collection) ............................................................. 615
Insert Method (StringList Collection) ............................................................... 616
Remove Method (StringList Collection) ............................................................. 617

Chapter 5
Xpedition Designer Schematic Editor Enumerated Types ..................................... 619
Xpedition Designer Enumerated Types Summary ............................................... 621
PinMappingType Enum ....................................................................................... 624
PropertyMappingType Enum .............................................................................. 625
ScopeReplaceSymbol Enum .............................................................................. 626
VdAllOrSelected Enum ....................................................................................... 627
VdAnnoObject Enum ......................................................................................... 628
VdAnnoPos Enum ............................................................................................... 629
VdAppEventDispatchID Enum ............................................................................. 630
VdArcPoint Enum .............................................................................................. 634
VdArrowType Enum ........................................................................................... 635
VdBoolean Enum ............................................................................................... 636
VdBusOrWire Enum ......................................................................................... 637
VdCorner Enum ............................................................................................... 638
Table of Contents

- VdCreateTime Enum .................................................. 639
- VdDataType Enum ...................................................... 640
- VdDocumentAccess Enum ............................................. 641
- VdFillStyle Enum ..................................................... 642
- VdFont Enum .......................................................... 644
- VdJointType Enum ..................................................... 646
- VdLabelVisibility Enum .............................................. 648
- VdLineCap Enum ....................................................... 649
- VdLineJoin Enum ...................................................... 650
- VdLinePattern Enum .................................................. 651
- VdLineStyle Enum ..................................................... 652
- VdNameType Enum ..................................................... 653
- VdNotifyFlag Enum ................................................... 654
- VdObjectClass Enum .................................................. 656
- VdObjectType Enum .................................................... 657
- VdObjectTypeMask Enum ............................................. 659
- VdOrigin Enum ........................................................ 664
- VdOrientation Enum ................................................... 663
- VdOpenMode Enum ..................................................... 662
- VdParamMode Enum .................................................... 665
- VdParamValue Enum ................................................... 669
- VdPinEndType Enum ................................................... 672
- VdRasterop Enum ....................................................... 673
- VdScope Enum ........................................................ 674
- VdSegmentEndType Enum ............................................. 675
- VdSelectionType Enum ............................................... 676
- VdSense Enum ........................................................ 677
- VdSheetSize Enum ..................................................... 678
- VdSide Enum .......................................................... 681
- VdSilentMode Enum .................................................... 682
- VdSourceDocumentType Enum ...................................... 683
- VdSplineOrder Enum .................................................. 684
- VdSplineType Enum ................................................... 685
- VdSymbolType Enum ................................................... 686
- VdTextFlags Enum ..................................................... 687
- VdUpdateOOScope Enum ............................................. 688
- VdUpdateOtherObjects Enum ....................................... 689
- VdVisibilityFlag Enum .............................................. 690
- VdWhichJoint Enum .................................................... 691

Chapter 6
Scripting with DataBook ............................................ 693

- DataBook Objects .................................................... 694
- Attribute Object ....................................................... 695
  - Name Property (Attribute Object) ................................ 696
  - NameVisible Property (Attribute Object) ...................... 697
  - Value Property (Attribute Object) ............................. 698
Appendix A
Changes to Xpedition Designer Automation .................................................. 723

Changes to Objects ................................................................. 724
Removed Objects ................................................................. 724
Changes to the Application Object .................................................. 724
Changes to the Block Object ....................................................... 728
Changes to the Arc Object ........................................................ 728
Changes to the Attribute Object .................................................... 729
Changes to the Box Object ........................................................ 729
Changes to the Circle Object ....................................................... 730
Changes to the Component Object ............................................... 730
Changes to the Connection Object ................................................. 730
Changes to the Label Object ....................................................... 731
Changes to the Line Object ........................................................ 731
Changes to the Net Object ........................................................ 732
Changes to the Pin Object ........................................................ 732
Changes to the Ripper Object ....................................................... 732
Changes to the Text Object ....................................................... 733
Changes to the Viewport Object .................................................... 733
Changes to Enumerated Types ...................................................... 734
Removed Enumerated Types ....................................................... 734
Changes to the VdAppEventDispatchID Enumerated Type .................. 734
Changes to the VdDocumentAccess Enumerated Type ....................... 735
Table of Contents

Changes to the VdNotifyFlag Enumerated Type ........................................... 736
Changes to the VdObjectClass Enumerated Type ........................................... 737
Changes to the VdObjectType Enumerated Type ........................................... 737
Changes to the VdObjectTypeMask Enumerated Type ................................. 738

End-User License Agreement
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 .......................................................... 19
Before You Begin ................................................................................. 20
Compatible Scripting Languages ............................................................. 23
Running Scripts .................................................................................. 24
  Search Order for Scripts .................................................................. 24
  The scripts.ini File ......................................................................... 25
  Debugging Scripts .......................................................................... 27
  Specifying Script and Form Execution .............................................. 29
Key Binding ......................................................................................... 32
  Key Binding Definition File ............................................................ 33

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
Before You Begin

- 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 1-1 on page 21.
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 1-2 on page 22.
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 PCB 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**

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 1-1. Special characters scripts.ini Files

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[application]</td>
<td>Square brackets must be used to enclose the name of the application to which the scripts apply.</td>
</tr>
<tr>
<td>*</td>
<td>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.</td>
</tr>
<tr>
<td>;</td>
<td>The semicolon is the comment character. Any text that follows a semicolon is considered comment text until a new line is located.</td>
</tr>
<tr>
<td>%variable_name%</td>
<td>Environmental variables can be used in scripts. For example, $HOME can be used to identify the home directory.</td>
</tr>
<tr>
<td>$variable_name</td>
<td></td>
</tr>
</tbody>
</table>

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.
Automation and Scripting

The scripts.ini File

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 debugger 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

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

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 Project Startup

You can execute a script or form when you open a specific project within 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.

**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 for this project, 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.

---

**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:

```
mgcscript MyClient.vbs [parameters]
```

The `mgcscript` host will also run script clients written in JScript.

---

**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 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 1-2

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Keybindings Enabled</th>
<th>Keybindings not Enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>exped_wvo.vbs</td>
<td>vdbindings.vbs</td>
</tr>
<tr>
<td>UNIX/LINUX</td>
<td>exped_pv.vbs</td>
<td>vdbindings.vbs</td>
</tr>
</tbody>
</table>

**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 Xpedition Designer 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

<table>
<thead>
<tr>
<th>Example</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example 1: Create/Add Menus</td>
<td>35</td>
</tr>
<tr>
<td>Example 2: Opening a Data Sheet</td>
<td>38</td>
</tr>
<tr>
<td>Example 3: Opening a Data Sheet</td>
<td>40</td>
</tr>
<tr>
<td>Example 4: Using Objects in Scripts</td>
<td>46</td>
</tr>
</tbody>
</table>

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 24.
Examples of Xpedition Designer Automation

Example 1: Create/Add Menus

---

**Note**

Whenever you create a custom menu with a script, be sure to include the following line in the script:

```plaintext
Scripting.DontExit = True
```

This line is necessary to prevent Xpedition Designer from crashing when the script is executed.

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.
Example 1: Create/Add Menus

```vba
Scripting.DontExit = True
' Insert a new 6th menu in Sheet Menu Bar. When the sheet is opened, 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 = "&Data Sheet"
' Add a menu item to the menu
Set button = menu.Controls.Add
' Menu item text
button.Caption = "View Data Sheet"
' 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 data sheet for the selected component.
Set menu = CommandBars("Component Popup").Controls.Add(cmdControlPopup,,,2)
' Menu text
menu.Caption = "&Data Sheet"
' Add a submenu
Set button = menu.Controls.Add ' Add a button to the menu
' Submenu item text
button.Caption = "View Data Sheet"
' 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 data sheet 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 Data Sheet"
' 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
```
Examples of Xpedition Designer Automation

Example 2: Opening a Data Sheet

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

This example launches Acrobat Reader to open the datasheet. 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 datasheets are located. The script then finds the DataSheet attribute value of the selected component and launches the datasheet in Acrobat.

In order for this script to run, a part 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 24.

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

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 DataSheet 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,"Data Sheet 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",,"Data Sheet 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
    ' Check to see if the DATAASHEET attribute has a value.
    If DataSheetName = "" Then
        MsgBox "Data sheet name not specified for component " &
    RefDesValue,","Data Sheet?"
    Else
        MsgBox "Refdes is: " & RefDesValue & vbCrLf &

Examples of Xpedition Designer Automation

Example 3: Opening a Data Sheet

The script in this section demonstrates how to open a data sheet.

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 24.
Examples of Xpedition Designer Automation

Example 3: Opening a Data Sheet

---

**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 fsf, 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

' Setting 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\ 
    D XD2007.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
        ' Create the View object if the document exists.
    End If ' OpenMySchematic
End If ' IsEmpty vddoc
Else
    ' Create the View object if the document exists.
    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 Xcel file with the same data as in the .txt file
Dim xl, ActSheet, ActCell, wb
Examples of Xpedition Designer Automation

Example 3: Opening a Data Sheet

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

Examples of Xpedition Designer Automation

Example 4: Using Objects in Scripts

' 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 Objects 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 an 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
- vddocColl - Xpedition Designer documents (open files in Xpedition Designer)
- vddoc - Xpedition Designer document (the active document)
- 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 24.

**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.
Examples of Xpedition Designer Automation

Example 4: Using Objects in Scripts

Option Explicit

Dim VDM_COMP, VD_ALL, VDFILL_VERT, VD_SELECTED
Dim vdapp, vddoc, vddf, 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 vddf = 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 = vddf.Block

' Create a variable whose value is the filename and extension.
Examples of Xpedition Designer Automation

Example 4: Using Objects in Scripts

CurBlockNamePage = CurBlock.GetName(1) & "." & CurBlock.SheetNum
MsgBox "The current schematic and page is : ", "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 : ", "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 : ", "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 setting
  ' 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.
Examples of Xpedition Designer Automation
Example 4: Using Objects in Scripts

Set gBox = wForm.Boxes(1)
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"
Examples of Xpedition Designer Automation

Example 4: Using Objects in Scripts

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 minimize/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 "_
Examples of Xpedition Designer Automation

Example 4: Using Objects in Scripts

& 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>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddinInfo Object</td>
<td>The AddinInfo object defines the characteristics necessary to load an addin into the application.</td>
</tr>
<tr>
<td>Application Object</td>
<td>The Application object represents Xpedition Designer application itself. Generally, it is the starting point for working with the automation capabilities of the application.</td>
</tr>
<tr>
<td>Arc Object</td>
<td>This object represents the Xpedition Designer arc graphical object.</td>
</tr>
<tr>
<td>Attribute Object</td>
<td>This object represents an attribute of an object (a net, component, block, or other object) on a schematic.</td>
</tr>
<tr>
<td>Block Object</td>
<td>This object represents a single sheet of a schematic or symbol.</td>
</tr>
<tr>
<td>Box Object</td>
<td>The Box is a graphical object in Xpedition Designer.</td>
</tr>
<tr>
<td>CColor Object</td>
<td>The CColor object represents a color in RGB format.</td>
</tr>
<tr>
<td>Circle Object</td>
<td>This object represents a graphical circle on a schematic.</td>
</tr>
<tr>
<td>CommandsManager Object</td>
<td>The CommandsManager object allows the user to perform command-related actions: (un)register, execute, enable/disable commands.</td>
</tr>
<tr>
<td>Component Object</td>
<td>This object represents a component placed on a schematic.</td>
</tr>
<tr>
<td>ComponentPin Object</td>
<td>This object represents a pin associated with a component on a schematic.</td>
</tr>
<tr>
<td>Connection Object</td>
<td>A connection object represents a component-to-net, net-to-bus, or bus-to-bus connection on a schematic.</td>
</tr>
<tr>
<td>HDLSourceDocument Object</td>
<td>This object represents an HDL source file document that has been opened in Xpedition Designer.</td>
</tr>
<tr>
<td>Label Object</td>
<td>The label object represents the label of an object in Xpedition Designer.</td>
</tr>
</tbody>
</table>
### Table 3-1. Xpedition Designer Objects (cont.)

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line Object</td>
<td>The Line Object represents a graphical line in a schematic.</td>
</tr>
<tr>
<td>Net Object</td>
<td>This object represents a net or bus on a schematic.</td>
</tr>
<tr>
<td>PDBPartitions Object</td>
<td>This object allows the user to manipulate PDB partitions data in a Xpedition Designer project file.</td>
</tr>
<tr>
<td>Pin Object</td>
<td>This object represents a symbol pin on a schematic.</td>
</tr>
<tr>
<td>Point Object</td>
<td>This object represents a point on a schematic specified by its X,Y coordinates.</td>
</tr>
<tr>
<td>ProjectData Object</td>
<td>ProjectData is an automation object which allows managing data stored in Xpedition Designer project file and provides some basic file-related information.</td>
</tr>
<tr>
<td>Rect Object</td>
<td>This object represents a graphic rectangle on a schematic, specified by its coordinates.</td>
</tr>
<tr>
<td>Ripper Object</td>
<td>This object represents a bus ripper on a schematic.</td>
</tr>
<tr>
<td>SchematicSheetDocument Object</td>
<td>This object represents one sheet of a schematic.</td>
</tr>
<tr>
<td>Segment Object</td>
<td>This object represents a portion of a net or bus on a schematic.</td>
</tr>
<tr>
<td>SymbolPartitions Object</td>
<td>This object allows you to manipulate symbol partitions data in a project file.</td>
</tr>
<tr>
<td>Text Object</td>
<td>This object represents text on a Schematic.</td>
</tr>
<tr>
<td>View Object</td>
<td>This object encapsulates the graphics display of a document (either a schematic or a symbol).</td>
</tr>
<tr>
<td>Viewport Object</td>
<td>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.</td>
</tr>
</tbody>
</table>
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 3-2. AddinInfo Object Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InitiallyDisabled Property</td>
<td>Specifies whether or not the addin is disabled when the host application is invoked.</td>
</tr>
<tr>
<td>InitiallyVisible Property</td>
<td>Specifies whether or not the addin is visible when the host application is invoked.</td>
</tr>
<tr>
<td>LicenseFeature Property</td>
<td>Gets or sets the license feature of the addin.</td>
</tr>
<tr>
<td>Name Property</td>
<td>Returns or sets the name of the addin.</td>
</tr>
<tr>
<td>Placement Property</td>
<td>Returns or sets the placement of the addin.</td>
</tr>
<tr>
<td>ProgId Property</td>
<td>Returns or sets the program identification of the addin.</td>
</tr>
<tr>
<td>RuntimeCreateDecision Property</td>
<td>Indicates whether or not the creation decision must be delegated to the module that hosts the addin.</td>
</tr>
<tr>
<td>ShortCutKey Property</td>
<td>Returns or sets the shortcut key associated with the addin.</td>
</tr>
<tr>
<td>ToolbarButton Property</td>
<td>Indicates whether or not the addin must have a toolbar button.</td>
</tr>
</tbody>
</table>
Schematic Editor Data Objects

**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”.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.
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>
<thead>
<tr>
<th>Method, Property, or Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activate Method (Application Object)</td>
<td>Makes the application window the active window.</td>
</tr>
<tr>
<td>AddAddin Method (Application Object)</td>
<td>Adds an addin to the application.</td>
</tr>
<tr>
<td>AppendOutput Method (Application Object)</td>
<td>Adds text to the specified tab of the Output window. If the tab does not already exist, the application creates it.</td>
</tr>
<tr>
<td>BroadcastDBConfigModified Method (Application Object)</td>
<td>Broadcast when AppEventDBCONFIG_MODIFIED occurs.</td>
</tr>
<tr>
<td>CloseProject Method (Application Object)</td>
<td>Closes the application project.</td>
</tr>
<tr>
<td>CommandsManager Method (Application Object)</td>
<td>Allows access to the CommandsManager object.</td>
</tr>
<tr>
<td>DesignComponents Method (Application Object)</td>
<td>Builds a collection of component objects for a schematic or an entire design.</td>
</tr>
<tr>
<td>DesignNets Method (Application Object)</td>
<td>Builds a collection of net objects for a schematic or an entire design.</td>
</tr>
<tr>
<td>DesignPaths Method (Application Object)</td>
<td>Builds a collection of component paths for the entire design.</td>
</tr>
<tr>
<td>GetActiveDesign Method (Application Object)</td>
<td>Retrieves the name of the active design.</td>
</tr>
<tr>
<td>GetDefaultColor Method (Application Object)</td>
<td>Returns the color (in RGB format) for the specified object type.</td>
</tr>
<tr>
<td>GetProjectData Method (Application Object)</td>
<td>Returns the ProjectData object.</td>
</tr>
</tbody>
</table>
### Table 3-3. Application Object Methods, Properties, and Events (cont.)

<table>
<thead>
<tr>
<th>Method, Property, or Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize Method</td>
<td>Initializes the application.</td>
</tr>
<tr>
<td>NewProject Method</td>
<td>Creates a new Xpedition Designer project. The new project folder must not already exist.</td>
</tr>
<tr>
<td>OpenBlocks Method</td>
<td>Returns a collection of Block objects.</td>
</tr>
<tr>
<td>OpenProject Method</td>
<td>Opens a project.</td>
</tr>
<tr>
<td>OpenURL Method</td>
<td>Opens a document at the location specified by a URL in the application.</td>
</tr>
<tr>
<td>ParamGetMode Method</td>
<td>Returns the global mode setting that controls how Xpedition Designer operates.</td>
</tr>
<tr>
<td>ParamGetValue Method</td>
<td>Returns the global value setting that controls how Xpedition Designer operates.</td>
</tr>
<tr>
<td>ParamSetMode Method</td>
<td>Sets the global mode setting that controls how Xpedition Designer operates.</td>
</tr>
<tr>
<td>ParamSetValue Method</td>
<td>Sets the global value setting that controls how Xpedition Designer operates.</td>
</tr>
<tr>
<td>PrintProject Method</td>
<td>Prints an entire design, or a portion of a design from the top level down to the level specified.</td>
</tr>
<tr>
<td>PushPath Method</td>
<td>Opens a schematic sheet by pushing through the components of a path.</td>
</tr>
<tr>
<td>Query Method</td>
<td>Returns a collection of objects based on type.</td>
</tr>
<tr>
<td>QueryPages Method</td>
<td>Returns a collection of block objects.</td>
</tr>
<tr>
<td>Quit Method</td>
<td>Terminates the application.</td>
</tr>
<tr>
<td>RunISE Method</td>
<td>Opens an existing non-local symbol in the embedded symbol editor. Otherwise, creates a new empty symbol with a given name.</td>
</tr>
<tr>
<td>SelectPath Method</td>
<td>Opens a schematic and selects objects based on a path from the top of the design.</td>
</tr>
<tr>
<td>Method, Property, or Event</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>SelectPathCompPin Method (Application Object)</td>
<td>Opens a schematic and selects a component pin based on a path from the top of the design.</td>
</tr>
<tr>
<td>SetDefaultColor Method (Application Object)</td>
<td>Sets the new color (in RGB format) for specified object type.</td>
</tr>
<tr>
<td>SetRedraw Method (Application Object)</td>
<td>Allows or prevents changes from being drawn.</td>
</tr>
<tr>
<td>ActiveDocument Property (Application Object)</td>
<td>Returns the ActiveDocument collection.</td>
</tr>
<tr>
<td>ActiveView Property (Application Object)</td>
<td>Returns the ActiveView collection for the currently activated schematic or symbol.</td>
</tr>
<tr>
<td>Addins Property (Application Object)</td>
<td>Allows access to an Addins collection.</td>
</tr>
<tr>
<td>CommandBars Property (Application Object)</td>
<td>Returns a collection of CommandBar objects.</td>
</tr>
<tr>
<td>CommandLineArguments Property (Application Object)</td>
<td>Returns the command line arguments that were used to call the application.</td>
</tr>
<tr>
<td>Interactive Property (Application Object)</td>
<td>Controls the response of the application to activity on its user interface.</td>
</tr>
<tr>
<td>QueueSelectEvents Property (Application Object)</td>
<td>Sets or disables event queuing.</td>
</tr>
<tr>
<td>ShellCmd Property (Application Object)</td>
<td>Returns the ShellCmd collection, which allows you to execute programs.</td>
</tr>
<tr>
<td>SilentMode Property (Application Object)</td>
<td>Returns or sets the silent mode status of the application.</td>
</tr>
<tr>
<td>StatusBarText Property (Application Object)</td>
<td>Returns or sets the text that is displayed in the application status bar.</td>
</tr>
<tr>
<td>Version Property (Application Object)</td>
<td>Returns the version number of the application that appears when you choose About Xpedition Designer on the Help menu.</td>
</tr>
</tbody>
</table>
### Table 3-3. Application Object Methods, Properties, and Events (cont.)

<table>
<thead>
<tr>
<th>Method, Property, or Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visible Property (Application Object)</td>
<td>Returns or sets the visibility of the application.</td>
</tr>
<tr>
<td>ActivateView Event (Application Object)</td>
<td>Occurs when the View child window becomes active.</td>
</tr>
<tr>
<td>ActivateView2 Event (Application Object)</td>
<td>Occurs when the View child window becomes active (receives focus), not when the application becomes active.</td>
</tr>
<tr>
<td>AfterDocumentOpened Event (Application Object)</td>
<td>Occurs after a document is opened and drawn.</td>
</tr>
<tr>
<td>AfterPrintProject Event (Application Object)</td>
<td>Occurs after a print project is completed.</td>
</tr>
<tr>
<td>AfterSheetRead Event (Application Object)</td>
<td>Occurs after a sheet is read into memory.</td>
</tr>
<tr>
<td>AfterSheetReRead Event (Application Object)</td>
<td>Occurs after a sheet has been reread.</td>
</tr>
<tr>
<td>BeforeDocumentOpened Event (Application Object)</td>
<td>Occurs before a document is opened and drawn.</td>
</tr>
<tr>
<td>BeforePrintProject Event (Application Object)</td>
<td>Occurs before a print project is executed.</td>
</tr>
<tr>
<td>BeforeProjectChanged Event (Application Object)</td>
<td>Occurs before the project is changed. That is, the event occurs before a opening another project.</td>
</tr>
<tr>
<td>BlockLocked Event (Application Object)</td>
<td>Occurs when you change the state of a schematic block from read-only to read/write, using the “Click to Edit” button.</td>
</tr>
<tr>
<td>BlockModified Event (Application Object)</td>
<td>Occurs every time a Block Object is modified.</td>
</tr>
<tr>
<td>CreateObject Event (Application Object)</td>
<td>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.</td>
</tr>
<tr>
<td>DeactivateView Event (Application Object)</td>
<td>Occurs when a child window is deactivated. This event is not triggered when the Application is deactivated.</td>
</tr>
<tr>
<td>DeactivateView2 Event (Application Object)</td>
<td>Occurs when a child window is deactivated. This event is not triggered by deactivating the Application.</td>
</tr>
<tr>
<td>Delete Event (Application Object)</td>
<td>Occurs whenever an object is deleted.</td>
</tr>
<tr>
<td>DocumentClose Event (Application Object)</td>
<td>Occurs whenever a SchematicSheetDocument Object is closed.</td>
</tr>
</tbody>
</table>
### Table 3-3. Application Object Methods, Properties, and Events (cont.)

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

\[ \text{Application}.\text{AddAddin}(\text{ByVal } p\text{AddinInfo As IAddinInfo}) \text{ As Boolean} \]

Arguments

- \( p\text{AddinInfo} \)
  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

For Help, press F1.
BroadcastDBConfigModified Method (Application Object)

Prerequisites: None
Object: Application Object
Broadcast when AppEventDBCONFIG_MODIFIED occurs.

Usage

Application.BroadcastDBConfigModified(ByVal sSource As String,
                          ByVal sPath As String) As Nothing

Arguments

- **sSource**
  A string that containing the name of the source.

- **sPath**
  A string containing the path.
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.
Schematic Editor Data Objects

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**

```vbnet
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.
Schematic Editor Data Objects

DesignNets Method (Application Object)

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 hierarchy.

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.
DesignSearcher Method (Application Object)

Prerequisites: None
Object: Application Object
DesignSearcher method of the application object. Returns IDesignSearcherAutomation interface.

Usage
Application.DesignSearcher() As IDesignSearcherAutomation

Arguments
None

Return Values
IDesignSearcherAutomation.
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 657.

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

Initializes 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.
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

\[ \text{Application} . \text{OpenProject}(\text{ByVal ProjectPath As String}) \]

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**

```vbnet
Application.OpenURL(ByVal URL As String)
```

**Arguments**

- **URL**  
  String that defines the URL for the document to be opened.
Schematic Editor Data Objects

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

\[ \text{Application.ParamGetValue}(\text{ByVal Index As VdParamMode}) \text{ As Long} \]

Arguments

- Index
  
  Index of the value of interest, as defined by VdParamMode 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*.
Schematic Editor Data Objects
ParamSetValue Method (Application Object)

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, ""
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(Flags As VdObjectTypeMask, 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.

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:
  o The path to the project file and the symbol (Netlist flow)
  o 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.2.3\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(Val 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.

- 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:
   o 0 - selects a component.
   o 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 657.

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

```vbnet
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


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


Description

CommandBar objects are the primary object to use when programming menus and toolbars (refer to Figure 3-1).
Figure 3-1. CommandBarServer Data Model

Legend
- Collection
- Method
- Read Property
- Object(Child)
- Method
- Read Property
- Read/Write Property
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.
Schematic Editor Data Objects

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.
Schematic Editor Data Objects
ShellCmd Property (Application Object)

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

\[
\text{Application} . \text{StatusBarText} = \text{String}
\]

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.

```vba
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.
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_ProtectChanged(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 cross probing.

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.
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 3-4. Arc Object Methods and Properties**

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

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 157.
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(Val 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(Val 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 254.
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>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete Method (Attribute Object)</td>
<td>Deletes the attribute.</td>
</tr>
<tr>
<td>DeleteInstanceValue Method (Attribute Object)</td>
<td>Deletes the instance value from the attribute.</td>
</tr>
<tr>
<td>GetLocation Method (Attribute Object)</td>
<td>Returns the coordinates of the attribute taking the form of a Point Object.</td>
</tr>
<tr>
<td>GetOatFull Method (Attribute Object)</td>
<td>Gets an instance value from within a particular context in the design.</td>
</tr>
<tr>
<td>GetObjectColor Method (Attribute Object)</td>
<td>Gets the color in which the attribute is drawn.</td>
</tr>
<tr>
<td>IsColorAutomatic Method (Attribute Object)</td>
<td>Determines if the attribute has an automatic color set.</td>
</tr>
<tr>
<td>SetAutomaticColor Method (Attribute Object)</td>
<td>Sets or unsets the color of an attribute object as the automatic color for attributes.</td>
</tr>
<tr>
<td>SetLocation Method (Attribute Object)</td>
<td>Specifies the location for an attribute.</td>
</tr>
<tr>
<td>SetObjectColor Method (Attribute Object)</td>
<td>Sets the color in which the Attribute is drawn.</td>
</tr>
<tr>
<td>Application Property (Attribute Object)</td>
<td>Returns the Application Object.</td>
</tr>
<tr>
<td>Child Property (Attribute Object)</td>
<td>Returns the child Block Object of the attribute.</td>
</tr>
<tr>
<td>EitherValue Property (Attribute Object)</td>
<td>Returns or sets the value of the attribute.</td>
</tr>
<tr>
<td>Font Property (Attribute Object)</td>
<td>Returns or sets the font that is used to draw the attribute.</td>
</tr>
<tr>
<td>InstanceValue Property (Attribute Object)</td>
<td>Returns or sets the instance (OAT) value for an attribute.</td>
</tr>
</tbody>
</table>
### Table 3-5. Attribute Object Methods and Properties (cont.)

<table>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name Property (Attribute Object)</td>
<td>Returns the name of the attribute.</td>
</tr>
<tr>
<td>Orientation Property (Attribute Object)</td>
<td>Returns or sets the attribute orientation.</td>
</tr>
<tr>
<td>Origin Property (Attribute Object)</td>
<td>Returns or sets one of the nine origins that is used for this attribute.</td>
</tr>
<tr>
<td>Parent Property (Attribute Object)</td>
<td>Returns the parent Block Object for the attribute.</td>
</tr>
<tr>
<td>Selected Property (Attribute Object)</td>
<td>Sets the selection status of the attribute.</td>
</tr>
<tr>
<td>Size Property (Attribute Object)</td>
<td>Returns or sets the text size for the attribute.</td>
</tr>
<tr>
<td>TextString Property (Attribute Object)</td>
<td>Returns or sets the string associated with an attribute.</td>
</tr>
<tr>
<td>Type Property (Attribute Object)</td>
<td>Returns the attribute type.</td>
</tr>
<tr>
<td>Value Property (Attribute Object)</td>
<td>Returns or sets the attribute's regular value even if the attribute has an instance (OAT) value.</td>
</tr>
<tr>
<td>Visible Property (Attribute Object)</td>
<td>Returns or sets the visibility of the attribute.</td>
</tr>
</tbody>
</table>
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

```vfp
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 173.
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

\textit{Attribute.SetAutomaticColor(byVal \textit{bAutomatic} as Boolean)}

Arguments

- \textit{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 \textit{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

\[ \text{Attribute.SetLocation}(\text{ByVal } X \text{ As Long, ByVal } Y \text{ As Long}) \]

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 180), otherwise EitherValue applies to the regular value (refer to “Value Property (Attribute Object)” on page 189).
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

\[ \text{Attribute.Font} = \text{VdFont} \]

Arguments

None

Return Values

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

_InstanceValue Property (Attribute Object)_

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

\[ \text{Attribute.Selected} = \text{True} \mid \text{False} \]

Arguments

None

Return Values

True \mid 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

\[ \text{Attribute.Value} = \text{String} \]

Arguments

None

Return Values

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

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

Examples

\[
\begin{align*}
\text{attr.Value} &= "U3" \\
\text{MsgBox} \ & \ \text{attr.Value}
\end{align*}
\]
Visible Property (Attribute Object)

Scope: Schematic editor
Object: Attribute Object
Access: Read/Write
Prerequisites: None
Returns or sets the visibility of the attribute.

Usage

\[ \text{Attribute.} \text{Visible} = \text{VdVisibilityFlag} \]

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 3-6. Block Object Methods and Properties

<table>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddArc Method (Block Object)</td>
<td>Adds an arc to the block.</td>
</tr>
<tr>
<td>AddAttribute Method (Block Object)</td>
<td>Adds an unattached attribute to the block.</td>
</tr>
<tr>
<td>AddBatchAttributes Method (Block Object)</td>
<td>Adds multiple unattached attributes on the active block as an encoded string.</td>
</tr>
<tr>
<td>AddBox Method (Block Object)</td>
<td>Creates a Box Object within the block.</td>
</tr>
<tr>
<td>AddCircle Method (Block Object)</td>
<td>Adds a circle to the block.</td>
</tr>
<tr>
<td>AddFub Method (Block Object)</td>
<td>Adds a functional block (FUB) to the block.</td>
</tr>
<tr>
<td>AddLine Method (Block Object)</td>
<td>Adds a line to the block.</td>
</tr>
<tr>
<td>AddLine2 Method (Block Object)</td>
<td>Adds a line to the block.</td>
</tr>
<tr>
<td>AddNet Method (Block Object)</td>
<td>Adds a net to the block.</td>
</tr>
<tr>
<td>AddNetEx Method (Block Object)</td>
<td>Adds a net to the block.</td>
</tr>
<tr>
<td>AddPartInstance Method (Block Object)</td>
<td>Adds a part instance to the block.</td>
</tr>
<tr>
<td>AddPin Method (Block Object)</td>
<td>Adds a pin to a symbol block.</td>
</tr>
<tr>
<td>AddPinAtLocation Method (Block Object)</td>
<td>Adds a pin to a symbol block.</td>
</tr>
<tr>
<td>AddSymbolInstance Method (Block Object)</td>
<td>Adds a symbol instance to the block.</td>
</tr>
<tr>
<td>AddText Method (Block Object)</td>
<td>Adds annotation text string.</td>
</tr>
</tbody>
</table>
### Table 3-6. Block Object Methods and Properties (cont.)

<table>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ApplySymbolUpdate Method (Block Object)</strong></td>
<td>Updates components with changes from their symbols.</td>
</tr>
<tr>
<td><strong>ChangeBorder Method (Block Object)</strong></td>
<td>Changes from one border symbol to another on a schematic.</td>
</tr>
<tr>
<td><strong>ChangeComponent Method (Block Object)</strong></td>
<td>Exchanges one or more components with a different component.</td>
</tr>
<tr>
<td><strong>ChangeComponentPreserve Refdes Method (Block Object)</strong></td>
<td>Exchanges one or more components with a different component and preserves the REFDES.</td>
</tr>
<tr>
<td><strong>ClearHighlight Method (Block Object)</strong></td>
<td>Marks components as current and clears the highlight.</td>
</tr>
<tr>
<td><strong>DeleteBorder Method (Block Object)</strong></td>
<td>Deletes a border symbol from a schematic. The border need not be selected, although it can be.</td>
</tr>
<tr>
<td><strong>DeleteSelected Method (Block Object)</strong></td>
<td>Deletes all currently selected objects.</td>
</tr>
<tr>
<td><strong>DeSelectAll Method (Block Object)</strong></td>
<td>Clears selection of all objects on this block.</td>
</tr>
<tr>
<td><strong>FindAttribute Method (Block Object)</strong></td>
<td>Searches for an attribute by its name.</td>
</tr>
<tr>
<td><strong>GetBatchAttributes Method (Block Object)</strong></td>
<td>Returns the unattached attributes on the currently active block as an encoded string.</td>
</tr>
<tr>
<td><strong>GetBboxPoint Method (Block Object)</strong></td>
<td>Returns the location of a specified corner of a bounding box.</td>
</tr>
<tr>
<td><strong>GetChildBlock Method (Block Object)</strong></td>
<td>Allows access to the child block object.</td>
</tr>
<tr>
<td><strong>GetName Method (Block Object)</strong></td>
<td>Returns the hierarchical path to the active view in the current block.</td>
</tr>
<tr>
<td><strong>InsertBorder Method (Block Object)</strong></td>
<td>Inserts a border symbol onto a schematic.</td>
</tr>
<tr>
<td><strong>PromoteSymbolNumbers Method (Block Object)</strong></td>
<td>Updates components with pin number changes from their symbols.</td>
</tr>
<tr>
<td><strong>RepositionAttributesAsOnSymbol Method (Block Object)</strong></td>
<td>Updates components with position changes from the symbol.</td>
</tr>
<tr>
<td><strong>SetZSheetSize Method (Block Object)</strong></td>
<td>Specifies width and height for Z sized blocks.</td>
</tr>
</tbody>
</table>
## Table 3-6. Block Object Methods and Properties (cont.)

<table>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UpdateBorder Method (Block Object)</td>
<td>Updates attribute values on a border symbol.</td>
</tr>
<tr>
<td>Application Property (Block Object)</td>
<td>Returns the Application Object.</td>
</tr>
<tr>
<td>Attributes Property (Block Object)</td>
<td>Returns a collection of Attribute Objects for the block.</td>
</tr>
<tr>
<td>DataType Property (Block Object)</td>
<td>Returns a value that relates whether the block is a schematic or a symbol.</td>
</tr>
<tr>
<td>IsFub Property (Block Object)</td>
<td>Returns a value that indicates whether the block is a functional block (FUB) or not.</td>
</tr>
<tr>
<td>LibraryName Property (Block Object)</td>
<td>Returns the library alias assigned to this block.</td>
</tr>
<tr>
<td>OpenMode Property (Block Object)</td>
<td>Returns a value that indicates how the Block was opened.</td>
</tr>
<tr>
<td>Parent Property (Block Object)</td>
<td>Returns the parent View Object of the block.</td>
</tr>
<tr>
<td>SheetNum Property (Block Object)</td>
<td>Returns the page number of the block.</td>
</tr>
<tr>
<td>SheetSize Property (Block Object)</td>
<td>Returns or sets the block drawing sheet size.</td>
</tr>
<tr>
<td>SymbolType Property (Block Object)</td>
<td>Returns or sets the symbol type used for the block.</td>
</tr>
<tr>
<td>Type Property (Block Object)</td>
<td>Returns the type for the block.</td>
</tr>
<tr>
<td>UpdateOutline Method (Block Object)</td>
<td>Updates the outline.</td>
</tr>
</tbody>
</table>
AddArc Method (Block Object)

Scope: Schematic editor
Object: Block Object
Prerequisites: None
Add 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(Val 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:
  o 0 = Invisible
  o 1 = Name and value visible
  o 2 = Name only visible
  o 3 = Value only visible

  <duplicate> should be one of the following integers:
  o 0 = add - add new attribute regardless of duplicates
  o 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\rl 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

\[ Block.AddFub(FubName As String, LowerLeftx As Long, LowerLefty As Long, UpperRightx As Long, UpperRighty As Long) As IVdComp \]

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
    Starting X location.
  - Locationy1
    Starting Y location.
  - Locationx2
    Ending X location.
  - Locationy2
    Ending Y location.
  - CompPin1
    Component Pin 1. May be NULL if not used.
  - CompPin2
    Component Pin 2. May be NULL if not used.
  - 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

You can add a net to a block between:

- Two locations X1,Y1 X2,Y2.
- Two component pins CompPin1 CompPin2.
- Any combination of Location X,Y and Component Pin.

Examples

The following example shows various combinations of the AddNet method.

```vba
Dim NullPin

Set NullPin=Nothing ' Makes a NULL object

'Adds Wire Net from (100,100) to (0,0)
Set N1 = ActiveView.Block.AddNet(100,100,0,0,NullPin,NullPin,VD_WIRE)

'Adds Wire Net from (100,100) to Cp1
Set N2 = ActiveView.Block.AddNet(100,100,0,0,Cp1,NullPin,VD_WIRE)

'Add Wire net between two component pins Cp1,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

- Locationx1
  Starting X location.
- Locationy1
  Starting Y location.
- Locationx2
  Ending X location.
- Locationy2
  Ending Y location.
- CompPin1
  Component Pin 1. May be NULL if not used.
- CompPin2
  Component Pin 2. May be NULL if not used.
- 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.

Note
All coordinates are measured in 100ths of an inch.
Description
You can add a net to a block between:

- Two locations X1,Y1 X2,Y2.
- Two component pins CompPin1 CompPin2.
- Any combination of Location X,Y and Component Pin.

Examples
The following example shows various combinations of the AddNetEx method.

```vba
Dim NullPin
Set NullPin=Nothing ' Makes a NULL object
'Adds Wire Net from (100,100) to (0,0)
Set N1 = ActiveView.Block.AddNetEx(100,100,0,0,NullPin,NullPin,VD_WIRE)

'Adds Wire Net from (100,100) to Cp1
Set N2 = ActiveView.Block.AddNetEx(100,100,0,0,Cp1,NullPin,VD_WIRE)

'Add Wire net between two component pins Cp1,Cp2
Set N3 = ActiveView.Block.AddNetEx(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

- 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

• 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.

Note

All coordinates are measured in 100ths of an inch.
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 or 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.

```vba
ActiveView.Block.ChangeComponentPreserveRefdes "< Selected Components >", "0805cap.1"
'dReplace 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(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

```java
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

\[ \text{Block.DeSelectAll()} \]

Arguments
None
FindAttribute Method (Block Object)

Scope: Schematic editor
Object: Block Object
Prerequisites: None
Searches for an attribute by its name.

Usage

\[ \text{Block.FindAttribute(ByVal AttributeName As String) As IVdAttr} \]

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

\[ \text{Block.SetZSheetSize}(\text{ByVal Width As Long, ByVal Height As Long}) \]

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.

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.
Schematic Editor Data Objects

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

 BlockSheetSize = 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 3-7. Box Object Methods and Properties

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

\[ \text{Box.SetObjectColor}(\text{ByVal newColor as IColor}) \]

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.
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>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>b Property (CColor Object)</code></td>
<td>Returns or sets the B (blue) component of color in RGB format.</td>
</tr>
<tr>
<td><code>g Property (CColor Object)</code></td>
<td>Returns or sets the G (green) component of color in RGB format. The value for green is 94.</td>
</tr>
<tr>
<td><code>r Property (CColor Object)</code></td>
<td>Returns or sets the R (red) component of color in RGB format.</td>
</tr>
</tbody>
</table>
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

\[ CColor.b = \text{Long} \]

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>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetCenter Method (Circle Object)</td>
<td>Returns the coordinates of the center of the circle.</td>
</tr>
<tr>
<td>GetObjectColor Method (Circle Object)</td>
<td>Gets the color in which the circle is drawn.</td>
</tr>
<tr>
<td>IsColorAutomatic Method (Circle Object)</td>
<td>Determines if the circle has been assigned an automatic color.</td>
</tr>
<tr>
<td>SetAutomaticColor Method (Circle Object)</td>
<td>Sets or unsets the color of a circle object as the automatic color for circles.</td>
</tr>
<tr>
<td>SetCenter Method (Circle Object)</td>
<td>Sets the coordinates of the center of a circle.</td>
</tr>
<tr>
<td>SetObjectColor Method (Circle Object)</td>
<td>Sets the color in which the circle is drawn.</td>
</tr>
<tr>
<td>Application Property (Circle Object)</td>
<td>Returns the Application Object of the circle.</td>
</tr>
<tr>
<td>FillStyle Property (Circle Object)</td>
<td>Returns or sets the fill style in which the circle is drawn.</td>
</tr>
<tr>
<td>LineStyle Property (Circle Object)</td>
<td>Returns or sets the line style in which the circle is drawn.</td>
</tr>
<tr>
<td>Parent Property (Circle Object)</td>
<td>Returns the parent Block Object for the circle.</td>
</tr>
<tr>
<td>Radius Property (Circle Object)</td>
<td>Returns or sets the radius for the circle.</td>
</tr>
<tr>
<td>Selected Property (Circle Object)</td>
<td>Sets the selection status of the circle.</td>
</tr>
<tr>
<td>Type Property (Circle Object)</td>
<td>Returns the type for the circle.</td>
</tr>
</tbody>
</table>
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.
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 262.
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.
SetCenter Method (Circle Object)

Scope: Schematic editor
Object: Circle Object
Prerequisites: None
Sets the coordinates of the center of a circle.

Usage

\[ \text{Circle.SetCenter}(\text{ByVal } X \text{ As Long, ByVal } Y \text{ As Long}) \]

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.
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**

```plaintext
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

\[ \text{Circle}. \text{LineStyle} = \text{VdLineStyle} \]

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.
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 3-10. CommandsManager Object Methods

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

\[ \text{CommandsManager.CommandRemove} \text{(ByVal CommandName As String)} \text{ As Boolean} \]

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.

```cpp
void CMyApp::OnExecuteCommand(LPDISPATCH pViewDrawApp, LPCTSTR Command, LPCTSTR RestOfLine)
{
}
```

VBSCRIPT

```vb
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
Revises a previously registered command.

Usage

\emph{CommandsManager.UnregisterOLECommand}(ByVal \emph{CommandName} As String, ByVal \emph{ClientDispatch} As Object) As Boolean

Arguments

- \emph{CommandName}
  This is the unique name for the command.

- \emph{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>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddAttribute Method (Component Object)</td>
<td>Adds an attribute to the component at the coordinates passed.</td>
</tr>
<tr>
<td>AddBatchAttributes Method (Component Object)</td>
<td>Adds multiple attributes to the component as an encoded string.</td>
</tr>
<tr>
<td>AddBatchOats Method (Component Object)</td>
<td>Adds multiple instance values (formerly known as OATs) to the component as an encoded string.</td>
</tr>
<tr>
<td>AddLabel Method (Component Object)</td>
<td>Adds a label to the component at the coordinates passed.</td>
</tr>
<tr>
<td>AddOat Method (Component Object)</td>
<td>Sets the instance value (formerly referred to as an OAT) for a component attribute.</td>
</tr>
<tr>
<td>FindAttribute Method (Component Object)</td>
<td>Searches all component attributes for one matching \textit{Name}.</td>
</tr>
<tr>
<td>GetBatchAttributes Method (Component Object)</td>
<td>Returns all attributes on the component as an encoded string.</td>
</tr>
<tr>
<td>GetBatchOats Method (Component Object)</td>
<td>Returns all instance values (OATs) on the component as an encoded string.</td>
</tr>
<tr>
<td>GetBboxPoint Method (Component Object)</td>
<td>Returns the coordinates of a specified corner of a bounding box.</td>
</tr>
<tr>
<td>GetConnections Method (Component Object)</td>
<td>Returns a collection of Connection Objects associated with component.</td>
</tr>
<tr>
<td>GetLocation Method (Component Object)</td>
<td>Returns the location coordinates (X,Y) for the component.</td>
</tr>
<tr>
<td>GetName Method (Component Object)</td>
<td>Returns the hierarchical path to the component.</td>
</tr>
<tr>
<td>SetLocation Method (Component Object)</td>
<td>Specifies the location for the component.</td>
</tr>
<tr>
<td>Application Property (Component Object)</td>
<td>Returns the Application Object.</td>
</tr>
<tr>
<td>Attributes Property (Component Object)</td>
<td>Returns a collection of Attribute Objects representing the attributes attached to this component.</td>
</tr>
</tbody>
</table>
### Table 3-11. Component Object Methods and Properties (cont.)

<table>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id Property (Component Object)</td>
<td>Returns a unique identifying integer for this component.</td>
</tr>
<tr>
<td>Label Property (Component Object)</td>
<td>Returns a label object.</td>
</tr>
<tr>
<td>Orientation Property (Component Object)</td>
<td>Returns or sets the component orientation (rotation).</td>
</tr>
<tr>
<td>Parent Property (Component Object)</td>
<td>Returns the parent Block Object of the component.</td>
</tr>
<tr>
<td>Refdes Property (Component Object)</td>
<td>Returns or sets the PCB reference designator for the component.</td>
</tr>
<tr>
<td>Scale Property (Component Object)</td>
<td>Returns or sets the scale factor of the component.</td>
</tr>
<tr>
<td>Selected Property (Component Object)</td>
<td>Sets the selection status of the component.</td>
</tr>
<tr>
<td>SymbolBlock Property (Component Object)</td>
<td>Returns the symbol which represents the component instance.</td>
</tr>
<tr>
<td>Type Property (Component Object)</td>
<td>Returns the type for the component.</td>
</tr>
<tr>
<td>UID Property (Component Object)</td>
<td>Returns the unique identifying string (UID) of this component.</td>
</tr>
</tbody>
</table>
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 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 to the component.

```c
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(Val 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:
  o 0 = Invisible
  o 1 = Name and Value Visible
  o 2 = Name Only Visible
  o 3 = Value Only Visible
  
  <duplicate> should be one of the following integers:
  o 0 = replace - if attribute with this name exists, replace it.
  o 1 = add - add new attribute regardless of duplicates.
  
  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 1 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 180, 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 attribute 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 638.

Return Values
As IVdPoint. The Point Object representing the coordinates.

Examples
This example displays both coordinates.

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
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. $111/$112) 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 $<\text{sheet number}>I<\text{ID number}> (for example: ($I34)).
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>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddAttribute Method (ComponentPin Object)</td>
<td>Adds an attribute to the component pin at the coordinates passed.</td>
</tr>
<tr>
<td>AddOAT Method (ComponentPin Object)</td>
<td>Sets the instance value (OAT) of a component pin attribute.</td>
</tr>
<tr>
<td>FindAttribute Method (ComponentPin Object)</td>
<td>Searches all component pin attributes for one matching AttributeName.</td>
</tr>
<tr>
<td>GetLocation Method (ComponentPin Object)</td>
<td>Returns a point object containing component pin coordinates.</td>
</tr>
<tr>
<td>Application Property (ComponentPin Object)</td>
<td>Returns the Application Object.</td>
</tr>
<tr>
<td>Attributes Property (ComponentPin Object)</td>
<td>Returns a collection of all Attribute Objects on this component pin whether they are visible or not.</td>
</tr>
<tr>
<td>Component Property (ComponentPin Object)</td>
<td>Returns the Component Object associated with the component pin.</td>
</tr>
<tr>
<td>Connection Property (ComponentPin Object)</td>
<td>Returns the Connection Object associated with the component pin.</td>
</tr>
<tr>
<td>Number Property (ComponentPin Object)</td>
<td>Returns or sets the component PCB pin number identifier.</td>
</tr>
<tr>
<td>Parent Property (ComponentPin Object)</td>
<td>Returns the parent Component Object for the component pin object.</td>
</tr>
<tr>
<td>Pin Property (ComponentPin Object)</td>
<td>Returns the Pin Object associated with the component pin.</td>
</tr>
<tr>
<td>Selected Property (ComponentPin Object)</td>
<td>Sets the selection status for the component pin.</td>
</tr>
<tr>
<td>Side Property (ComponentPin Object)</td>
<td>Returns the side of the component on which the component pin exists.</td>
</tr>
<tr>
<td>Type Property (ComponentPin Object)</td>
<td>Returns the component pin type.</td>
</tr>
</tbody>
</table>
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 AttributeName.

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 attribute 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

\[
\text{ComponentPin}.\text{Attributes}
\]

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.
Schematic Editor Data Objects

Connection Property (ComponentPin Object)

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>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompPin Property (Connection Object)</td>
<td>Returns the ComponentPinObject for this connection.</td>
</tr>
<tr>
<td>Net Property (Connection Object)</td>
<td>Returns the Net Object for the connection.</td>
</tr>
<tr>
<td>Ripper Property (Connection Object)</td>
<td>Returns the Ripper Object for the connection.</td>
</tr>
<tr>
<td>Segment Property (Connection Object)</td>
<td>Returns the Segment Object associated with this connection.</td>
</tr>
</tbody>
</table>
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.
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 3-14. HDLSourceDocument Object Methods and Properties

<table>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BookmarkLine Method (HDLSourceDocument Object)</td>
<td>Adds a bookmark to a specified line in the HDL document.</td>
</tr>
<tr>
<td>GotoLine Method (HDLSourceDocument Object)</td>
<td>Moves the cursor to a specified line in the text editor.</td>
</tr>
<tr>
<td>Name Property (HDLSourceDocument Object)</td>
<td>Sets or returns the name of the HDLSourceDocument Object.</td>
</tr>
<tr>
<td>Path Property (HDLSourceDocument Object)</td>
<td>Sets or returns the path of the HDLSourceDocument Object.</td>
</tr>
</tbody>
</table>
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.
Schematic Editor Data Objects

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 3-15. Label Object Methods and Properties

<table>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLocation Method (Label Object)</td>
<td>Returns the coordinates of the label as a Point Object.</td>
</tr>
<tr>
<td>GetObjectColor Method (Label Object)</td>
<td>Gets the color in which the label is drawn.</td>
</tr>
<tr>
<td>IsColorAutomatic Method (Label Object)</td>
<td>Determines if the label has an automatic color set for it.</td>
</tr>
<tr>
<td>SetAutomaticColor Method (Label Object)</td>
<td>Sets or unsets the color of a label object as the automatic color for labels.</td>
</tr>
<tr>
<td>SetLocation Method (Label Object)</td>
<td>Specifies the coordinates for the label.</td>
</tr>
<tr>
<td>SetObjectColor Method (Label Object)</td>
<td>Sets the color in which the label is drawn.</td>
</tr>
<tr>
<td>Application Property (Label Object)</td>
<td>Returns the Application Object.</td>
</tr>
<tr>
<td>Font Property (Label Object)</td>
<td>Returns or sets the font used for the label.</td>
</tr>
<tr>
<td>Orientation Property (Label Object)</td>
<td>Returns or sets the orientation for the label.</td>
</tr>
<tr>
<td>Origin Property (Label Object)</td>
<td>Returns or sets the coordinates for the origin of the label.</td>
</tr>
<tr>
<td>Parent Property (Label Object)</td>
<td>Returns the parent object for the label (the object on which the label is placed).</td>
</tr>
<tr>
<td>ResolvedName Property (Label Object)</td>
<td>Returns the text string of the label and fully resolves any nicknames.</td>
</tr>
<tr>
<td>Scope Property (Label Object)</td>
<td>Returns or sets the scope of the label.</td>
</tr>
<tr>
<td>Selected Property (Label Object)</td>
<td>Sets the selection status for the label.</td>
</tr>
<tr>
<td>Sense Property (Label Object)</td>
<td>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.</td>
</tr>
</tbody>
</table>
### Table 3-15. Label Object Methods and Properties (cont.)

<table>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Size Property (Label Object)</strong></td>
<td>Returns or sets the size (text height) of the label.</td>
</tr>
<tr>
<td><strong>TextString Property (Label Object)</strong></td>
<td>Returns or sets the text value of the label.</td>
</tr>
<tr>
<td><strong>Type Property (Label Object)</strong></td>
<td>Returns the type value VDTS_LABEL, indicating that the object is, in fact, a label.</td>
</tr>
<tr>
<td><strong>Visible Property (Label Object)</strong></td>
<td>Returns or sets the visibility of the label.</td>
</tr>
</tbody>
</table>
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.
Schematic Editor Data Objects

**GetObjectColor Method (Label Object)**

Scope: Schematic editor
Object: Label Object
Prerequisites: None

Gets the color in which the label is drawn.

**Usage**

\[
\text{ActiveColor} = \text{Label.GetObjectColor()} \text{ As IColor}
\]

**Arguments**

None

**Return Values**

*As Color*. See “CColor Object” on page 254.
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 338.
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 254.
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

\[ \text{Label.Font} = \text{VdFont} \]

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

\[ \text{Label}.\text{Sense} = \text{VdSense} \]

Arguments

None

Return Values

VdSense. The return/set type for this property of the form \text{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

\[ \text{Label.Visible} = \text{VdLabelVisibility} \]

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>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddPoint Method (Line Object)</td>
<td>Adds a new endpoint to the line.</td>
</tr>
<tr>
<td>GetNumPoints Method (Line Object)</td>
<td>Returns the number of points in the line.</td>
</tr>
<tr>
<td>GetObjectColor Method (Line Object)</td>
<td>Gets the color in which the line is drawn.</td>
</tr>
<tr>
<td>GetPoint Method (Line Object)</td>
<td>Returns the coordinates of the specified Point Object.</td>
</tr>
<tr>
<td>IsColorAutomatic Method (Line Object)</td>
<td>Determines if the line has an automatic color set for it.</td>
</tr>
<tr>
<td>SetAutomaticColor Method (Line Object)</td>
<td>Sets or unsets the color of a line object as the automatic color for lines.</td>
</tr>
<tr>
<td>SetObjectColor Method (Line Object)</td>
<td>Sets the color in which the line is drawn.</td>
</tr>
<tr>
<td>Application Property (Line Object)</td>
<td>Returns the Application Object.</td>
</tr>
<tr>
<td>LineStyle Property (Line Object)</td>
<td>Returns or sets the style for the line.</td>
</tr>
<tr>
<td>Parent Property (Line Object)</td>
<td>Returns the parent Attribute Object for the line.</td>
</tr>
<tr>
<td>Selected Property (Line Object)</td>
<td>Sets the selection status for the line.</td>
</tr>
<tr>
<td>Type Property (Line Object)</td>
<td>Returns the value VDTS_LINE, indicating that the object is a line.</td>
</tr>
</tbody>
</table>
AddPoint Method (Line Object)

Scope: Schematic editor
Object: Line Object
Prerequisites: None
Adds a new endpoint to the line.

Usage

```vbnet
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 254.
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

\textit{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 360.
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

\texttt{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 \texttt{IsColorAutomatic Method (Line Object)}, the actual 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

```vba
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 254.
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

\[ \text{Line.Selected} = \text{True} \mid \text{False} \]

Arguments

None

Return Values

True \mid \text{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>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddAttribute Method (Net Object)</td>
<td>Adds an Attribute Object to the net on the specified segment at the coordinates passed.</td>
</tr>
<tr>
<td>AddLabel Method (Net Object)</td>
<td>Adds a Label Object to the net.</td>
</tr>
<tr>
<td>Connections Method (Net Object)</td>
<td>Returns a collection of Connection Objects. Use this collection to access component pin, net and segment objects.</td>
</tr>
<tr>
<td>FindAttribute Method (Net Object)</td>
<td>Locates the attribute specified by the passed argument.</td>
</tr>
<tr>
<td>GetConnectedLabel Method (Net Object)</td>
<td>Gets the net label for the specified Segment Object, or any connected segments. This method is useful for identifying segments in a bus net.</td>
</tr>
<tr>
<td>GetConnectedNetName Method (Net Object)</td>
<td>Gets the net label for the specified Segment Object, or any connected segments. This method is useful to identify segments in a bus.</td>
</tr>
<tr>
<td>GetLabel Method (Net Object)</td>
<td>Gets the net label on the specified Segment Object.</td>
</tr>
<tr>
<td>GetObjectColor Method (Net Object)</td>
<td>Gets the color in which the net is drawn.</td>
</tr>
<tr>
<td>GetRippers Method (Net Object)</td>
<td>Returns the collection of rippers that are connected to a bus.</td>
</tr>
<tr>
<td>GetSegments Method (Net Object)</td>
<td>Returns a collection of the segments that make up a net.</td>
</tr>
<tr>
<td>GetSignals Method (Net Object)</td>
<td>Returns a list of constituent bus signals.</td>
</tr>
<tr>
<td>GetSingleJointLocs Method (Net Object)</td>
<td>Returns an encoded string of the coordinates of all single joints (dangle boxes).</td>
</tr>
<tr>
<td>IsColorAutomatic Method (Net Object)</td>
<td>Determines if the net has an automatic color set for it.</td>
</tr>
<tr>
<td>IsSegmentSelected Method (Net Object)</td>
<td>Determines if a net segment is selected.</td>
</tr>
</tbody>
</table>
Table 3-17. Net Object Methods and Properties (cont.)

<table>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SelectSegment Method (Net Object)</td>
<td>Selects the specified Segment Object.</td>
</tr>
<tr>
<td>SelectSegmentByJointLoc Method (Net Object)</td>
<td>Selects a Segment Object based on the location and type of the lower joint in the segment.</td>
</tr>
<tr>
<td>SetAutomaticColor Method (Net Object)</td>
<td>Sets or unsets the color of a net object as the automatic color for nets.</td>
</tr>
<tr>
<td>SetObjectColor Method (Net Object)</td>
<td>Sets the color in which the net is drawn.</td>
</tr>
<tr>
<td>Application Property (Net Object)</td>
<td>Returns the Application Object.</td>
</tr>
<tr>
<td>Attributes Property (Net Object)</td>
<td>Returns a collection object which contains all attributes on this net whether they are visible or not.</td>
</tr>
<tr>
<td>Id Property (Net Object)</td>
<td>Returns a unique identifying integer for the net.</td>
</tr>
<tr>
<td>LineStyle Property (Net Object)</td>
<td>Returns or sets the line style for the net.</td>
</tr>
<tr>
<td>Parent Property (Net Object)</td>
<td>Returns the parent Block Object in which this net was created.</td>
</tr>
<tr>
<td>Selected Property (Net Object)</td>
<td>Sets the selection status for the net.</td>
</tr>
<tr>
<td>Type Property (Net Object)</td>
<td>Returns VDTS_NET, indicating that the object is, in fact, a net.</td>
</tr>
<tr>
<td>UID Property (Net Object)</td>
<td>Returns a Unique Identifying String (UID) for the net.</td>
</tr>
</tbody>
</table>
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(Val 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 254.
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

\[ Net.GetSingleJointLocs() \text{ As String} \]

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 385.
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

\[
\text{Net.SelectSegmentByJointLoc(ByVal XCoordinate As Long, ByVal YCoordinate As Long, ByVal JointType As VdJointType)}
\]

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

\[ \text{Net.SetObjectColor(ByVal newColor as IColor)} \]

Arguments

- newColor
  The color assigned to the net. The new color is assigned as a Color object, as described in “CColor Object” on page 254.
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.
Schematic Editor Data Objects
LineStyle Property (Net Object)

**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**

```plaintext
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

\[ \text{Net.Selected} = \text{True} \mid \text{False} \]

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 3-18. PDBPartitions Object Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AppendPDBPartition Method</strong> (PDBPartitions Object)</td>
<td>Adds a new entry to the end of the PDB partitions list for a given design.</td>
</tr>
<tr>
<td><strong>GetPDBPartition Method</strong> (PDBPartitions Object)</td>
<td>Returns the name of the PDB partition.</td>
</tr>
<tr>
<td><strong>GetPDBPartitionsArray Method</strong> (PDBPartitions Object)</td>
<td>Returns the list of PDB partitions for a given design.</td>
</tr>
<tr>
<td><strong>InsertPDBPartition Method</strong> (PDBPartitions Object)</td>
<td>Inserts a new entry into the PDB partitions list at the specified index position for a given design.</td>
</tr>
<tr>
<td><strong>PDBPartitionExists Method</strong> (PDBPartitions Object)</td>
<td>Checks to see if a PDB partition with a specific name exists in the PDB partition collection of a given design.</td>
</tr>
<tr>
<td><strong>RemovePDBPartitionByIndex Method</strong> (PDBPartitions Object)</td>
<td>Removes the PDB Partition located at the specified index position from the PDB partitions list of a given design.</td>
</tr>
<tr>
<td><strong>RemovePDBPartitionByName Method</strong> (PDBPartitions Object)</td>
<td>Removes the PDB Partition with the specified name from the PDB partitions list of a given design.</td>
</tr>
</tbody>
</table>
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, siCDBDesign 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

\[ \text{PDBPartitions.GetPDBPartition(ByVal Index As Long, ByVal siCDBDesign As String) As String} \]

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

\[ \text{PDBPartitions.RemovePDBPartitionByIndex(Val Index As Long, siCDBDesign As String)} \]

Arguments

- **Index**
  
  Index position of the partition to be removed from the PDB partition. Indexing starts at 1.

- **siCDBDesign**
  
  The name of the design.
Schematic Editor Data Objects
RemovePDBPartitionByName Method (PDBPartitions Object)

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 3-19. Pin Object Methods and Properties

<table>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddAttribute Method (Pin Object)</td>
<td>Adds an unattached attribute to the pin.</td>
</tr>
<tr>
<td>FindAttribute Method (Pin Object)</td>
<td>Finds an attribute on a pin.</td>
</tr>
<tr>
<td>GetLocation Method (Pin Object)</td>
<td>Returns the location of the pin.</td>
</tr>
<tr>
<td>GetName Method (Pin Object)</td>
<td>Returns the name of the pin.</td>
</tr>
<tr>
<td>GetObjectColor Method (Pin Object)</td>
<td>Gets the color in which the pin is drawn.</td>
</tr>
<tr>
<td>SetLocation Method (Pin Object)</td>
<td>Specifies the interior and exterior coordinates of the pin.</td>
</tr>
<tr>
<td>Application Property (Pin Object)</td>
<td>Returns the Application Object.</td>
</tr>
<tr>
<td>Attributes Property (Pin Object)</td>
<td>Returns a collection object which contains all attributes on this pin whether they are visible or not.</td>
</tr>
<tr>
<td>Id Property (Pin Object)</td>
<td>Returns a unique identifying integer for the pin.</td>
</tr>
<tr>
<td>Label Property (Pin Object)</td>
<td>Returns the Label Object associated with the pin.</td>
</tr>
<tr>
<td>Parent Property (Pin Object)</td>
<td>Returns the parent Block Object for this pin.</td>
</tr>
<tr>
<td>Selected Property (Pin Object)</td>
<td>Sets the selection status for the pin.</td>
</tr>
<tr>
<td>Sense Property (Pin Object)</td>
<td>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.</td>
</tr>
<tr>
<td>Side Property (Pin Object)</td>
<td>Returns the side of the component on which the pin exists.</td>
</tr>
<tr>
<td>Type Property (Pin Object)</td>
<td>Returns VDTS_PIN, indicating that the object is, in fact, a pin.</td>
</tr>
<tr>
<td>UID Property (Pin Object)</td>
<td>Returns a unique identifying string (UID) for this pin.</td>
</tr>
</tbody>
</table>
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**

```vbnet
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 254.
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

\texttt{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

\texttt{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

\[ Pin.\text{Selected} = \text{True} \mid \text{False} \]

Arguments

None

Return Values

True \mid 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

\[ \text{Pin.Sense} = \text{VdSense} \]

Arguments

None

Return Values

VdSense. The return/set type for this property. This is of the form \text{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

\( \text{Pin.Side} \)

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>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X Property (Point Object)</td>
<td>Returns or sets the X coordinate of a point object.</td>
</tr>
<tr>
<td>Y Property (Point Object)</td>
<td>Returns or sets the Y coordinate for a point object.</td>
</tr>
</tbody>
</table>
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

\[
Point.X = \text{Long}
\]

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 Y coordinate 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>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddiCDBDesign Method (ProjectData Object)</td>
<td>Adds new design data to a project file.</td>
</tr>
<tr>
<td>GetBordersFilePath Method (ProjectData Object)</td>
<td>Returns the path of the border symbols file.</td>
</tr>
<tr>
<td>GetBusContentsFilePath Method (ProjectData Object)</td>
<td>Returns the path of the bus contents file.</td>
</tr>
<tr>
<td>GetiCDBDesignRootBlock Method (ProjectData Object)</td>
<td>Returns the name of the top-level block for the given design.</td>
</tr>
<tr>
<td>GetiCDBDesigns Method (ProjectData Object)</td>
<td>Returns a collection of design names specified in the project file.</td>
</tr>
<tr>
<td>GetiCDBDesignType Method (ProjectData Object)</td>
<td>Returns the type of specified design.</td>
</tr>
<tr>
<td>GetiCDBDiscardFilePath Method (ProjectData Object)</td>
<td>Returns the path of the discard file.</td>
</tr>
<tr>
<td>GetPCBDesignPath Method (ProjectData Object)</td>
<td>Returns the path of the PCB file for a given design.</td>
</tr>
<tr>
<td>GetPDBPartitions Method (ProjectData Object)</td>
<td>Returns the PDBPartitions Object for the project.</td>
</tr>
</tbody>
</table>
### Table 3-21. ProjectData Object Methods and Properties (cont.)

<table>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetPinComponentsFilePath Method (ProjectData Object)</td>
<td>Returns the path of the pin components file.</td>
</tr>
<tr>
<td>GetProjectFilePath Method (ProjectData Object)</td>
<td>Returns the full file path of project file including project file name.</td>
</tr>
<tr>
<td>GetProjectName Method (ProjectData Object)</td>
<td>Returns the project file name, without extension.</td>
</tr>
<tr>
<td>GetProjectPath Method (ProjectData Object)</td>
<td>Returns the full file path of the project file, excluding the project file name.</td>
</tr>
<tr>
<td>GetSearchPathScheme Method (ProjectData Object)</td>
<td>Returns the library search path scheme for a given design.</td>
</tr>
<tr>
<td>GetSymbolPartitions Method (ProjectData Object)</td>
<td>Returns the SymbolPartitions Object for the project.</td>
</tr>
<tr>
<td>RemoveiCDBDesign Method (ProjectData Object)</td>
<td>Removes design data from a project file.</td>
</tr>
<tr>
<td>RenameiCDBDesign Method (ProjectData Object)</td>
<td>Changes the name for existing design data in the project file.</td>
</tr>
<tr>
<td>SetBordersFilePath Method (ProjectData Object)</td>
<td>Sets the path for the borders symbol file.</td>
</tr>
<tr>
<td>SetBusContentsFilePath Method (ProjectData Object)</td>
<td>Sets the path of the bus contents file.</td>
</tr>
<tr>
<td>SetiCDBDesignRootBlock Method (ProjectData Object)</td>
<td>Sets the name of the top-level block for a given design.</td>
</tr>
<tr>
<td>SetiCDBDesignType Method (ProjectData Object)</td>
<td>Sets the type for the specified design.</td>
</tr>
<tr>
<td>SetiCDBDiscardFilePath Method (ProjectData Object)</td>
<td>Sets the path for the discard file.</td>
</tr>
<tr>
<td>SetPCBDesignPath Method (ProjectData Object)</td>
<td>Sets the path of the PCB file for a given design.</td>
</tr>
</tbody>
</table>
### Table 3-21. ProjectData Object Methods and Properties (cont.)

<table>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetPinComponentsFilePath Method (ProjectData Object)</td>
<td>Sets the path for the pin components file.</td>
</tr>
<tr>
<td>SetSearchPathScheme Method (ProjectData Object)</td>
<td>Sets the library search path scheme for a given design.</td>
</tr>
<tr>
<td>UpdateOtherObjects Method (ProjectData Object)</td>
<td>Updates the specified objects for the project.</td>
</tr>
<tr>
<td>CentralLibraryPath Property (ProjectData Object)</td>
<td>Gets or sets the central library path.</td>
</tr>
<tr>
<td>iCDBDir Property (ProjectData Object)</td>
<td>Gets the iCDB database path.</td>
</tr>
</tbody>
</table>
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

\[ \text{ProjectData.GetBusContentsFilePath(ByVal resolveSoftPrefix As Boolean) As String} \]

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.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 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.
Schematic Editor Data Objects

GetPinComponentsFilePath Method (ProjectData Object)

Scope: Schematic editor
Object: ProjectData Object
Prerequisites: None
Returns the path of the pin components file.

Usage

\texttt{ProjectData.GetPinComponentsFilePath(ByVal resolveSoftPrefix As Boolean) As String}

Arguments

- \texttt{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 siBusContentFilePath As String)

Arguments

- siBusContentFilePath
  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.SetPinComponentFilePath(ByVal sPinComponentsFilePath As String)

Arguments

- sPinComponentFilePath
  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>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bottom Property (Rect Object)</td>
<td>Returns or sets the Y coordinate of the bottom side of the rectangle.</td>
</tr>
<tr>
<td>Left Property (Rect Object)</td>
<td>Returns or sets the X coordinate of the left side of the rectangle.</td>
</tr>
<tr>
<td>Right Property (Rect Object)</td>
<td>Returns or sets the X coordinate of the right side of the rectangle.</td>
</tr>
<tr>
<td>Top Property (Rect Object)</td>
<td>Returns or sets the Y coordinate of the top of the rectangle.</td>
</tr>
</tbody>
</table>
**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.
Schematic Editor Data Objects

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>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GetConnectedObject Method (Ripper Object)</strong></td>
<td>Returns the net or bus that is located at the other side of the bus ripper.</td>
</tr>
<tr>
<td><strong>GetConnectedObjects Method (Ripper Object)</strong></td>
<td>Returns the collection of Net Objects that are connected to that bus ripper.</td>
</tr>
<tr>
<td><strong>GetMappedSignal Method (Ripper Object)</strong></td>
<td>Returns the name of the signal that is mapped to the given signal by the bus ripper.</td>
</tr>
</tbody>
</table>
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 3-24. SchematicSheetDocument Object Methods and Properties**

<table>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activate Method (SchematicSheetDocument Object)</td>
<td>Assigns the first view of the Views collection owned by this document as the active view.</td>
</tr>
<tr>
<td>Close Method (SchematicSheetDocument Object)</td>
<td>Closes the document and flushes it from memory.</td>
</tr>
<tr>
<td>DiscardSymbolChanges Method (SchematicSheetDocument Object)</td>
<td>Discards local symbol without updating design or saving changes.</td>
</tr>
<tr>
<td>ExportMetafile Method (SchematicSheetDocument Object)</td>
<td>Exports the current document as a metafile for plotting.</td>
</tr>
<tr>
<td>GetViews Method (SchematicSheetDocument Object)</td>
<td>Returns a collection of View objects which contain this document.</td>
</tr>
<tr>
<td>IsReadOnly Method (SchematicSheetDocument Object)</td>
<td>Determines if the current document is read-only (locked).</td>
</tr>
<tr>
<td>Print Method (SchematicSheetDocument Object)</td>
<td>Prints a document range to the default printer.</td>
</tr>
<tr>
<td>ReRead Method (SchematicSheetDocument Object)</td>
<td>Causes the specified page of the current document to be reread from disk.</td>
</tr>
<tr>
<td>Save Method (SchematicSheetDocument Object)</td>
<td>Saves changes made in the embedded symbol editor to a non-local symbol.</td>
</tr>
<tr>
<td>SaveAs Method (SchematicSheetDocument Object)</td>
<td>Saves non-local symbol opened in the embedded symbol editor to a different name.</td>
</tr>
</tbody>
</table>
### Table 3-24. SchematicSheetDocument Object Methods and Properties (cont.)

<table>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UpdateSymbolInDesign Method</strong> (SchematicSheetDocument Object)</td>
<td>Updates symbol in the schematic to include the current local changes.</td>
</tr>
<tr>
<td><strong>Application Property</strong> (SchematicSheetDocument Object)</td>
<td>Returns the application property for the document.</td>
</tr>
<tr>
<td><strong>FullName Property</strong> (SchematicSheetDocument Object)</td>
<td>Returns the name of the schematic sheet.</td>
</tr>
<tr>
<td><strong>Name Property</strong> (SchematicSheetDocument Object)</td>
<td>Sets or returns the name of the schematic sheet.</td>
</tr>
<tr>
<td><strong>Parent Property</strong> (SchematicSheetDocument Object)</td>
<td>Returns the parent Attribute Object for the document.</td>
</tr>
</tbody>
</table>
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(From As Integer, To As Integer, 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

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

run_ESE = "-lmc C:\MentorGraphics\EEVX.2.3\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.2.3\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 3-25. Segment Object Methods and Properties

<table>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetJointType Method</td>
<td>Returns the type of a specified joint.</td>
</tr>
<tr>
<td>(Segment Object)</td>
<td>Examples: <a href="#">Link</a></td>
</tr>
<tr>
<td>IsBus Method (Segment Object)</td>
<td>Determines if the segment is a bus.</td>
</tr>
<tr>
<td>Location Method (Segment Object)</td>
<td>Returns the coordinates of specified joint in the form of a Point Object.</td>
</tr>
<tr>
<td>Application Property</td>
<td>Returns the Application Object.</td>
</tr>
<tr>
<td>(Segment Object)</td>
<td>Examples: <a href="#">Link</a></td>
</tr>
<tr>
<td>Attributes Property</td>
<td>Returns a collection object that contains all Attribute Objects on this segment, whether they are visible or not.</td>
</tr>
<tr>
<td>(Segment Object)</td>
<td>Examples: <a href="#">Link</a></td>
</tr>
<tr>
<td>Parent Property (Segment Object)</td>
<td>Returns the parent Net Object of the segment.</td>
</tr>
<tr>
<td>Type Property (Segment Object)</td>
<td>Returns VDTS_SEGMENT, indicating that the object is, in fact, a segment.</td>
</tr>
<tr>
<td></td>
<td>Examples: <a href="#">Link</a></td>
</tr>
</tbody>
</table>
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
Determine 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

```vbnet
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.
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>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppendSymbolPartition Method (SymbolPartitions Object)</td>
<td>Adds a new entry to the end of the symbol partitions list for a given design.</td>
</tr>
<tr>
<td>GetSymbolPartition Method (SymbolPartitions Object)</td>
<td>Returns the name of a symbol partition (specified by index) for a given design from the symbol partitions list.</td>
</tr>
<tr>
<td>GetSymbolPartitionsArray Method (SymbolPartitions Object)</td>
<td>Returns the list of symbol partitions for a given design.</td>
</tr>
<tr>
<td>GetSymbolPartitionsCount Method (SymbolPartitions Object)</td>
<td>Returns the number of symbol partitions for a given design.</td>
</tr>
<tr>
<td>InsertSymbolPartition Method (SymbolPartitions Object)</td>
<td>Inserts a new entry into the symbol partitions list for a design, at the specified position.</td>
</tr>
<tr>
<td>RemoveSymbolPartitionByIndex Method (SymbolPartitions Object)</td>
<td>Removes an entry (specified by index) from the symbol partitions list for a given design.</td>
</tr>
<tr>
<td>RemoveSymbolPartitionByName Method (SymbolPartitions Object)</td>
<td>Removes an entry (specified by name) from the symbol partitions list for a given design.</td>
</tr>
<tr>
<td>SymbolPartitionExists Method (SymbolPartitions Object)</td>
<td>Indicates whether the specified name exists in the symbol partitions list for a given design.</td>
</tr>
</tbody>
</table>
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.
Schematic Editor Data Objects

**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**

```vba
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.
Schematic Editor Data Objects
GetSymbolPartitionsCount Method (SymbolPartitions Object)

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, siCDBDesign 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>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetLocation Method (Text Object)</td>
<td>Returns the coordinates of the text object in the form of a Point Object.</td>
</tr>
<tr>
<td>GetObjectColor Method (Text Object)</td>
<td>Gets the color in which the text is drawn.</td>
</tr>
<tr>
<td>IsColorAutomatic Method (Text Object)</td>
<td>Determines if the text has an automatic color set for it.</td>
</tr>
<tr>
<td>SetAutomaticColor Method (Text Object)</td>
<td>Sets or unsets the color of a text object as the automatic color for text.</td>
</tr>
<tr>
<td>SetLocation Method (Text Object)</td>
<td>Specifies the coordinates of the text object.</td>
</tr>
<tr>
<td>SetObjectColor Method (Text Object)</td>
<td>Sets the color in which the text is drawn.</td>
</tr>
<tr>
<td>Application Property (Text Object)</td>
<td>Returns the Application Object.</td>
</tr>
<tr>
<td>Font Property (Text Object)</td>
<td>Returns or sets the font in which the text is written.</td>
</tr>
<tr>
<td>Orientation Property (Text Object)</td>
<td>Returns or sets the orientation of the text.</td>
</tr>
<tr>
<td>Origin Property (Text Object)</td>
<td>Returns or sets the origin for the text.</td>
</tr>
<tr>
<td>Parent Property (Text Object)</td>
<td>Returns the parent Block Object for the text.</td>
</tr>
<tr>
<td>Selected Property (Text Object)</td>
<td>Sets the selection status for the text.</td>
</tr>
<tr>
<td>Size Property (Text Object)</td>
<td>Returns or sets the size (height) of the text.</td>
</tr>
<tr>
<td>TextString Property (Text Object)</td>
<td>Returns or sets the text string; that is, the actual contents of the text.</td>
</tr>
<tr>
<td>Type Property (Text Object)</td>
<td>Returns VDTS_TEXT, indicating that the object is, in fact, text.</td>
</tr>
</tbody>
</table>
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 254.
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 501.
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 254.
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

\[ Text.Selected = \text{True} \mid \text{False} \]

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 124). 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>
<thead>
<tr>
<th>Method, Property, or Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activate Method (View Object)</td>
<td>Assigns the view from which this method is invoked as the active view.</td>
</tr>
<tr>
<td>AddAttributeMoveMode Method (View Object)</td>
<td>Adds an attribute to the currently active view attached to the cursor.</td>
</tr>
<tr>
<td>Application Method (View Object)</td>
<td>Returns the Application Object.</td>
</tr>
<tr>
<td>BufferCopy Method (View Object)</td>
<td>Copies the selected items to the buffer.</td>
</tr>
<tr>
<td>BufferCut Method (View Object)</td>
<td>Cuts the selected items to the buffer.</td>
</tr>
<tr>
<td>BufferPaste Method (View Object)</td>
<td>Pastes the selected items from the buffer.</td>
</tr>
<tr>
<td>BufferPasteXY Method (View Object)</td>
<td>Pastes the data from the buffer to the active view at the reference point specified. This method does not require user intervention.</td>
</tr>
<tr>
<td>ComputeMBB Method (View Object)</td>
<td>Sets the values of the four coordinate values to define the page bounding box.</td>
</tr>
<tr>
<td>Document Method (View Object)</td>
<td>Returns the ShematicSheetDocument Object contained by the view.</td>
</tr>
<tr>
<td>GetJointLocs Method (View Object)</td>
<td>Returns a string that contains the X,Y locations of the joints specified.</td>
</tr>
<tr>
<td>GetName Method (View Object)</td>
<td>Returns the hierarchical path to the block associated with this view.</td>
</tr>
<tr>
<td>GetSelectedNetName Method (View Object)</td>
<td>Returns the name of the selected net(s).</td>
</tr>
<tr>
<td>GetTopLevelDesignName Method (View Object)</td>
<td>Returns the block name associated with the top level hierarchical design for this view.</td>
</tr>
</tbody>
</table>
### View Object Methods, Properties, and Events (cont.)

<table>
<thead>
<tr>
<th>Method, Property, or Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ModifyVisibility Method (View Object)</td>
<td>Changes the visibility of the target attributes where the attribute name string matches the string <code>SelectString</code>.</td>
</tr>
<tr>
<td>Query Method (View Object)</td>
<td>Returns a collection of objects based on type.</td>
</tr>
<tr>
<td>Refresh Method (View Object)</td>
<td>Causes the application to refresh the graphical depiction of the view. Call this method after graphical changes, moves, pastes, copies or deletions.</td>
</tr>
<tr>
<td>SelectByName Method (View Object)</td>
<td>Selects an object by name.</td>
</tr>
<tr>
<td>SelectByName2 Method (View Object)</td>
<td>Selects objects by name.</td>
</tr>
<tr>
<td>SelectObject Method (View Object)</td>
<td>Selects an object by name.</td>
</tr>
<tr>
<td>SelectSegmentByJointLoc Method (View Object)</td>
<td>Selects a net segment based on the location and type of the lower joint in the segment.</td>
</tr>
<tr>
<td>SelectText Method (View Object)</td>
<td>Selects text strings that match the regular expression.</td>
</tr>
<tr>
<td>SetCenter Method (View Object)</td>
<td>Centers the view around the coordinates (maintaining the current zoom factor).</td>
</tr>
<tr>
<td>ViewFull Method (View Object)</td>
<td>Maximizes the view and redraws the graphical display.</td>
</tr>
<tr>
<td>ZoomIn Method (View Object)</td>
<td>Executes a zoom in.</td>
</tr>
<tr>
<td>ZoomOut Method (View Object)</td>
<td>Executes a zoom out.</td>
</tr>
<tr>
<td>ZoomSelect Method (View Object)</td>
<td>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.</td>
</tr>
<tr>
<td>Block Property (View Object)</td>
<td>Returns a Block Object contained within the view.</td>
</tr>
<tr>
<td>TopBlock Property (View Object)</td>
<td>Returns the Block Object associated with the top level hierarchical design for this view.</td>
</tr>
<tr>
<td>Viewport Property (View Object)</td>
<td>Returns the graphics Viewport Object.</td>
</tr>
</tbody>
</table>
Table 3-28. View Object Methods, Properties, and Events (cont.)

<table>
<thead>
<tr>
<th>Method, Property, or Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnActivate Event (View Object)</td>
<td>Occurs when a view is being activated.</td>
</tr>
<tr>
<td>OnSelect Event (View Object)</td>
<td>Occurs when an object on a view is selected.</td>
</tr>
</tbody>
</table>
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

\[ \text{View} . \text{ComputeMBB} (\text{ByVal OLEItems As Boolean, Xmin As Long, Ymin As Long, Xmax As Long, Ymax As Long}) \]

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.
**GetMethodName (View Object)**

Scope: Schematic editor

Object: View Object

Prerequisites: None

Returns the hierarchical path to the block associated with this view.

**Usage**

```csharp
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**

```vbnet
View.Query(Val 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 657.
- 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”:

```vbnet
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
```
Schematic Editor Data Objects

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

\texttt{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>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arc Method (Viewport Object)</td>
<td>Draws an arc, specified by three points.</td>
</tr>
<tr>
<td>Arrow Method (Viewport Object)</td>
<td>Draws an arrow from one point to another.</td>
</tr>
<tr>
<td>Box Method (Viewport Object)</td>
<td>Draws a box.</td>
</tr>
<tr>
<td>Circle Method (Viewport Object)</td>
<td>Draws a circle at the specified location with the specified radius.</td>
</tr>
<tr>
<td>Ellipse Method (Viewport Object)</td>
<td>Draws an ellipse centered at X,Y with radii XRadius, YRadius.</td>
</tr>
<tr>
<td>EraseRectangle Method (Viewport Object)</td>
<td>Erases a rectangle.</td>
</tr>
<tr>
<td>GetObjectColor Method (Viewport Object)</td>
<td>Gets the color in which the viewport is drawn.</td>
</tr>
<tr>
<td>Line Method (Viewport Object)</td>
<td>Draws a line from one point to another.</td>
</tr>
<tr>
<td>PixelRectangle Method (Viewport Object)</td>
<td>Returns the viewport rectangle in terms of pixel coordinates.</td>
</tr>
<tr>
<td>PixelToUser Method (Viewport Object)</td>
<td>Translates pixel coordinates into user coordinates.</td>
</tr>
<tr>
<td>Point Method (Viewport Object)</td>
<td>Draws a pixel at a specified location.</td>
</tr>
<tr>
<td>PolyLine Method (Viewport Object)</td>
<td>Draws a polygon line.</td>
</tr>
<tr>
<td>SetClipRectangle Method (Viewport Object)</td>
<td>Sets the clipping region for drawing in the form of a Rect Object.</td>
</tr>
<tr>
<td>SetObjectColor Method (Viewport Object)</td>
<td>Sets the color in which the viewport is drawn.</td>
</tr>
</tbody>
</table>
**Table 3-29. Viewport Methods and Properties (cont.)**

<table>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spline Method (Viewport Object)</td>
<td>Draws a spline with specified control points.</td>
</tr>
<tr>
<td>Text Method (Viewport Object)</td>
<td>Draws a text string at the specified location.</td>
</tr>
<tr>
<td>UserRectangle Method (Viewport Object)</td>
<td>Returns a viewport rectangle using the specified coordinates.</td>
</tr>
<tr>
<td>UserToPixel Method (Viewport Object)</td>
<td>Translates user coordinates into pixel coordinates.</td>
</tr>
<tr>
<td>FillStyle Property (Viewport Object)</td>
<td>Returns or sets the fill style for polygons, circles and boxes.</td>
</tr>
<tr>
<td>LineCap Property (Viewport Object)</td>
<td>Returns or sets the style of the end point of a line.</td>
</tr>
<tr>
<td>LineJoin Property (Viewport Object)</td>
<td>Returns or sets the line join type for polygons.</td>
</tr>
<tr>
<td>LinePattern Property (Viewport Object)</td>
<td>Returns or sets the line pattern type.</td>
</tr>
<tr>
<td>LineThickness Property (Viewport Object)</td>
<td>Returns or sets line thickness.</td>
</tr>
<tr>
<td>RasterMode Property (Viewport Object)</td>
<td>Returns or sets the raster mode.</td>
</tr>
<tr>
<td>TextAngle Property (Viewport Object)</td>
<td>Sets the text angle.</td>
</tr>
<tr>
<td>TextFont Property (Viewport Object)</td>
<td>Sets the font used for text.</td>
</tr>
<tr>
<td>TextSize Property (Viewport Object)</td>
<td>Sets the size (height) of the text.</td>
</tr>
</tbody>
</table>
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.
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 254.
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.
Schematic Editor Data Objects

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.

```vbnet
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(Left As Long, Top As Long, Right As Long, 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 254.
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 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

```vbnet
Viewport.Text(ByVal X As Long, ByVal Y As Long, ByVal String As String, ByVal Flags As VdTextFlags)
```

Arguments

- **X**
  - X coordinate.
- **Y**
  - Y coordinate.
- **String**
  - The actual text message that is to be drawn.
- **Flags**
  - Text flags, of the form `VdTextFlags Enum`.

**Note**

All coordinates are measured in 100ths of an inch.
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.
Schematic Editor Data Objects

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.
**Scope**: Schematic editor  
**Object**: Viewport Object  
**Access**: Read/Write  
**Prerequisites**: None  
Returns or sets the raster mode.

**Usage**

```plaintext
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

$Viewport\text{.TextAngle} = \text{Long}$

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.
Schematic Editor Data Objects

TextSize Property (Viewport Object)
Chapter 4  
Xpedition Designer Schematic Editor Object Collections

The following table includes summary information for each collection you can access in Xpedition Designer automation.

Table 4-1. Xpedition Designer Collections

<table>
<thead>
<tr>
<th>Collection</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDLSourceDocuments</td>
<td>This object represents a collection of HDLSourceDocument Objects.</td>
</tr>
<tr>
<td>Collection</td>
<td>This object represents a collection of HDLSourceDocument Objects.</td>
</tr>
<tr>
<td>SchematicSheetDocuments</td>
<td>Documents is an automation collection that represents the set of Xpedition</td>
</tr>
<tr>
<td>Collection</td>
<td>Designer schematic documents.</td>
</tr>
<tr>
<td>StringCollection</td>
<td>This object represents a collection of strings that exist on a Xpedition</td>
</tr>
<tr>
<td>Collection</td>
<td>Designer schematic. This collection provides methods for identifying and/or</td>
</tr>
<tr>
<td></td>
<td>removing strings from the schematic, but not otherwise modifying them.</td>
</tr>
<tr>
<td>StringList Collection</td>
<td>This object is a collection of strings that exist on a Xpedition Designer</td>
</tr>
<tr>
<td></td>
<td>schematic. This collection has a well-defined interface that allows for</td>
</tr>
<tr>
<td></td>
<td>removing/adding/iterating any of the elements that compose the collection.</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item Method (HDLSourceDocuments Collection)</td>
<td>Returns an HDLSourceDocument Object contained in the collection.</td>
</tr>
<tr>
<td>New Method (HDLSourceDocuments Collection)</td>
<td>Creates a new HDLSourceDocument Object based on the specified type, then adds it to this collection.</td>
</tr>
<tr>
<td>Open Method (HDLSourceDocuments Collection)</td>
<td>Opens an HDLSource Document Object derived from the document path and adds it to this collection.</td>
</tr>
<tr>
<td>Remove Method (HDLSourceDocuments Collection)</td>
<td>Removes a source file from the collection, using the index.</td>
</tr>
<tr>
<td>RemoveAll Method (HDLSourceDocuments Collection)</td>
<td>Removes all HDLSourceDocument Objects from the collection.</td>
</tr>
<tr>
<td>SaveAll Method (HDLSourceDocuments Collection)</td>
<td>Saves all HDL source documents in the HDLSourceDocuments Collection.</td>
</tr>
<tr>
<td>Count Property (HDLSourceDocuments Collection)</td>
<td>Returns the number of HDLSourceDocument Objects contained in the collection.</td>
</tr>
</tbody>
</table>
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


Examples

Using C++:

```c++
For Index=1 To HDLSourceDocuments.Count
    Set Doc = HDLSourceDocuments.Item(Index)
Next
```

Using VBScript:

```vb
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`. 

Note - Viewing PDF files within a web browser causes some links not to function. Use HTML for full navigation.
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 the `HDLSourceDocument 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>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close Method (SchematicSheetDocuments Collection)</td>
<td>Closes all the currently open documents.</td>
</tr>
<tr>
<td>CopyToClipboard Method (SchematicSheetDocuments Collection)</td>
<td>Copies sheets from a schematic to the clipboard.</td>
</tr>
<tr>
<td>DeleteSheet Method (SchematicSheetDocuments Collection)</td>
<td>Deletes a specified sheet from a specified schematic.</td>
</tr>
<tr>
<td>GetAvailableSchematics Method (SchematicSheetDocuments Collection)</td>
<td>Returns the collection of all schematic document names for the current project.</td>
</tr>
<tr>
<td>GetAvailableSheets Method (SchematicSheetDocuments Collection)</td>
<td>Returns the collection of sheet names for a given schematic document.</td>
</tr>
<tr>
<td>InsertSheet Method (SchematicSheetDocuments Collection)</td>
<td>Inserts a specified sheet into a specified schematic.</td>
</tr>
<tr>
<td>IsSymbolUnderEdit Method (SchematicSheetDocuments Collection)</td>
<td>Checks whether the symbol has local changes that have not been updated in the design.</td>
</tr>
<tr>
<td>Item Method (SchematicSheetDocuments Collection)</td>
<td>Returns a SchematicSheetDocument object contained in the collection.</td>
</tr>
<tr>
<td>Open Method (SchematicSheetDocuments Collection)</td>
<td>Opens an existing schematic document and adds it to the collection.</td>
</tr>
<tr>
<td>Open_Hierarchically Method (SchematicSheetDocuments Collection)</td>
<td>Opens an existing schematic document hierarchically and adds it to the collection.</td>
</tr>
<tr>
<td>Method or Property</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OpenSymbol Method (SchematicSheetDocuments Collection)</td>
<td>Opens the symbol of a given name to edit.</td>
</tr>
<tr>
<td>PasteFromClipboard Method (SchematicSheetDocuments Collection)</td>
<td>Pastes the contents of the clipboard into the specified schematic, possibly replacing sheets within the schematic.</td>
</tr>
<tr>
<td>Application Property (SchematicSheetDocuments Collection)</td>
<td>Returns the Application Object.</td>
</tr>
<tr>
<td>Count Property (SchematicSheetDocuments Collection)</td>
<td>Returns the number of objects contained in the collection.</td>
</tr>
<tr>
<td>Parent Property (SchematicSheetDocuments Collection)</td>
<td>Returns or the parent collection of the collection.</td>
</tr>
</tbody>
</table>
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.CopyToClipboard(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 4-1. CopyToClipboard](image-url)
CopyToClipboard Method (SchematicSheetDocuments Collection)

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.

```vbscript
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 4-2. 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)
Object: 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)

Collection: SchematicSheetDocuments Collection
Access: Read-Only
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

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

OpenSymbol Method (SchematicSheetDocuments Collection)

Scope: Schematic editor - Embedded symbol editor (ESE)
Object: 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
Paste the contents of the clipboard into the specified schematic, possibly replacing sheets within the schematic.

Usage

SchematicSheetDocuments.PasteFromClipboard(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>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item Method (StringCollection Collection)</td>
<td>Returns an element (specified by its index number) from the collection.</td>
</tr>
<tr>
<td>Remove Method (StringCollection Collection)</td>
<td>Removes an element (specified by its index number) from the collection.</td>
</tr>
<tr>
<td>Count Property (StringCollection Collection)</td>
<td>Returns the number of elements in the collection.</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Append Method (StringList Collection)</td>
<td>Adds an element to the end of the collection.</td>
</tr>
<tr>
<td>Clear Method (StringList Collection)</td>
<td>Removes all elements from the collection.</td>
</tr>
<tr>
<td>GetCount Method (StringList Collection)</td>
<td>Returns the number of elements in the collection.</td>
</tr>
<tr>
<td>GetItem Method (StringList Collection)</td>
<td>Returns a string (specified by Index) that is contained in the collection.</td>
</tr>
<tr>
<td>Insert Method (StringList Collection)</td>
<td>Inserts an element into the collection at the position specified by the index argument.</td>
</tr>
<tr>
<td>Remove Method (StringList Collection)</td>
<td>Removes an element specified by its index position from the collection.</td>
</tr>
</tbody>
</table>
Append Method (StringList Collection)

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

\[ \text{StringList.Insert(ByVal } s\text{Item As String, ByVal } Index \text{ As Long)} \]

Arguments

- \( s\text{Item} \)
  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.
Xpedition Designer Schematic Editor Object Collections

Remove Method (StringList Collection)
Chapter 5
Xpedition Designer Schematic Editor
Enumerated Types

The Xpedition Designer schematic editor supports a wide range of enumerated types.

Xpedition Designer Enumerated Types Summary

<table>
<thead>
<tr>
<th>Enum Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>PinMappingType Enum</td>
<td>624</td>
</tr>
<tr>
<td>PropertyMappingType Enum</td>
<td>625</td>
</tr>
<tr>
<td>ScopeReplaceSymbol Enum</td>
<td>626</td>
</tr>
<tr>
<td>VdAllOrSelected Enum</td>
<td>627</td>
</tr>
<tr>
<td>VdAnnoObject Enum</td>
<td>628</td>
</tr>
<tr>
<td>VdAnnoPos Enum</td>
<td>629</td>
</tr>
<tr>
<td>VdAppEventDispatchID Enum</td>
<td>630</td>
</tr>
<tr>
<td>VdArcPoint Enum</td>
<td>634</td>
</tr>
<tr>
<td>VdArrowType Enum</td>
<td>635</td>
</tr>
<tr>
<td>VdBoolean Enum</td>
<td>636</td>
</tr>
<tr>
<td>VdBusOrWire Enum</td>
<td>637</td>
</tr>
<tr>
<td>VdCorner Enum</td>
<td>638</td>
</tr>
<tr>
<td>VdCreateTime Enum</td>
<td>639</td>
</tr>
<tr>
<td>VdDataType Enum</td>
<td>640</td>
</tr>
<tr>
<td>VdDocumentAccess Enum</td>
<td>641</td>
</tr>
<tr>
<td>VdFillStyle Enum</td>
<td>642</td>
</tr>
<tr>
<td>VdFont Enum</td>
<td>644</td>
</tr>
<tr>
<td>VdJointType Enum</td>
<td>646</td>
</tr>
<tr>
<td>VdLabelVisibility Enum</td>
<td>648</td>
</tr>
<tr>
<td>VdLineCap Enum</td>
<td>649</td>
</tr>
<tr>
<td>VdLineJoin Enum</td>
<td>650</td>
</tr>
<tr>
<td>VdLinePattern Enum</td>
<td>651</td>
</tr>
<tr>
<td>VdLineStyle Enum</td>
<td>652</td>
</tr>
<tr>
<td>VdNameType Enum</td>
<td>653</td>
</tr>
<tr>
<td>VdNotifyFlag Enum</td>
<td>654</td>
</tr>
<tr>
<td>VdObjectClass Enum</td>
<td>656</td>
</tr>
<tr>
<td>VdObjectType Enum</td>
<td>657</td>
</tr>
<tr>
<td>VdObjectTypeMask Enum</td>
<td>659</td>
</tr>
<tr>
<td>VdOnOff Enum</td>
<td>661</td>
</tr>
<tr>
<td>VdOpenMode Enum</td>
<td>662</td>
</tr>
<tr>
<td>VdOrientation Enum</td>
<td>663</td>
</tr>
<tr>
<td>VdOrigin Enum</td>
<td>664</td>
</tr>
<tr>
<td>VdParamMode Enum</td>
<td>665</td>
</tr>
<tr>
<td>VdParamValue Enum</td>
<td>669</td>
</tr>
<tr>
<td>VdPinEndType Enum</td>
<td>672</td>
</tr>
</tbody>
</table>
Xpedition Designer Schematic Editor Enumerated Types

VdRasterop Enum ........................................ 673
VdScope Enum ........................................... 674
VdSegmentEndType Enum ................................. 675
VdSelectionType Enum ................................. 676
VdSense Enum ........................................... 677
VdSheetSize Enum ....................................... 678
VdSide Enum ............................................. 681
VdSilentMode Enum ..................................... 682
VdSourceDocumentType Enum ......................... 683
VdSplineOrder Enum .................................... 684
VdSplineType Enum ..................................... 685
VdSymbolType Enum ..................................... 686
VdTextFlags Enum ....................................... 687
VdUpdateOOScope Enum ................................. 688
VdUpdateOtherObjects Enum ......................... 689
VdVisibilityFlag Enum ................................. 690
VdWhichJoint Enum ..................................... 691
Xpedition Designer Enumerated Types Summary

The table below includes summary information for each enumerated type you can access with Xpedition Designer Automation.

<table>
<thead>
<tr>
<th>Enumerated Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PinMappingType Enum</td>
<td>Specifies a pin mapping type.</td>
</tr>
<tr>
<td>PropertyMappingType Enum</td>
<td>Specifies a property mapping type.</td>
</tr>
<tr>
<td>ScopeReplaceSymbol Enum</td>
<td>Specifies the replace symbol scope.</td>
</tr>
<tr>
<td>VdAllOrSelected Enum</td>
<td>Specifies whether to consider all items or only those that are currently selected.</td>
</tr>
<tr>
<td>VdAnnoObject Enum</td>
<td>Specifies the type of object to be annotated.</td>
</tr>
<tr>
<td>VdAnnoPos Enum</td>
<td>Specifies an origin point for an annotation.</td>
</tr>
<tr>
<td>VdAppEventDispatchID Enum</td>
<td>Specifies the condition that causes an event to occur.</td>
</tr>
<tr>
<td>VdArcPoint Enum</td>
<td>Specifies a point that defines an arc.</td>
</tr>
<tr>
<td>VdArrowType Enum</td>
<td>Specifies the type of arrowheads used for drawing a line.</td>
</tr>
<tr>
<td>VdBoolean Enum</td>
<td>Specifies the boolean value of an argument or condition.</td>
</tr>
<tr>
<td>VdBusOrWire Enum</td>
<td>Specifies whether a net is a bus or a wire.</td>
</tr>
<tr>
<td>VdCorner Enum</td>
<td>Specifies a corner point for an object.</td>
</tr>
<tr>
<td>VdCreateTime Enum</td>
<td>Specifies the time at which a notification appears when an object it placed.</td>
</tr>
<tr>
<td>VdDataType Enum</td>
<td>Specifies the block data type.</td>
</tr>
<tr>
<td>VdDocumentAccess Enum</td>
<td>Specifies the document access types.</td>
</tr>
<tr>
<td>VdFillStyle Enum</td>
<td>Specifies a fill style.</td>
</tr>
<tr>
<td>VdFont Enum</td>
<td>Specifies a font style.</td>
</tr>
<tr>
<td>VdJointType Enum</td>
<td>Specifies a joint type.</td>
</tr>
<tr>
<td>VdLabelVisibility Enum</td>
<td>Specifies visibility status.</td>
</tr>
<tr>
<td>VdLineCap Enum</td>
<td>Specifies the line cap used for lines.</td>
</tr>
<tr>
<td>VdLineJoin Enum</td>
<td>Specifies the style for a line join.</td>
</tr>
</tbody>
</table>
### Table 5-1. Xpedition Designer Enumerated Types (cont.)

<table>
<thead>
<tr>
<th>Enumerated Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VdLinePattern Enum</td>
<td>Specifies the pattern used for lines.</td>
</tr>
<tr>
<td>VdLineStyle Enum</td>
<td>Specifies the style used for lines.</td>
</tr>
<tr>
<td>VdNameType Enum</td>
<td>Specifies the format that is displayed for net names.</td>
</tr>
<tr>
<td>VdNotifyFlag Enum</td>
<td>Specifies the events that trigger flags.</td>
</tr>
<tr>
<td>VdObjectClass Enum</td>
<td>Specifies an object class.</td>
</tr>
<tr>
<td>VdObjectType Enum</td>
<td>Specifies an object type.</td>
</tr>
<tr>
<td>VdObjectTypeMask Enum</td>
<td>Specifies an object type mask (usually for selection purposes).</td>
</tr>
<tr>
<td>VdOnOff Enum</td>
<td>Specifies mode settings.</td>
</tr>
<tr>
<td>VdOpenMode Enum</td>
<td>Specifies the mode for an opened block.</td>
</tr>
<tr>
<td>VdOrientation Enum</td>
<td>Specifies an orientation style.</td>
</tr>
<tr>
<td>VdOrigin Enum</td>
<td>Specifies a point of origin.</td>
</tr>
<tr>
<td>VdParamMode Enum</td>
<td>Specifies various modes for the Xpedition Designer session.</td>
</tr>
<tr>
<td>VdParamValue Enum</td>
<td>Specifies various values used in the Xpedition Designer session.</td>
</tr>
<tr>
<td>VdPinEndType Enum</td>
<td>Specifies a pin end type.</td>
</tr>
<tr>
<td>VdRasterop Enum</td>
<td>Specifies raster operations.</td>
</tr>
<tr>
<td>VdScope Enum</td>
<td>Specifies the scope.</td>
</tr>
<tr>
<td>VdSegmentEndType Enum</td>
<td>Specifies the segment end type.</td>
</tr>
<tr>
<td>VdSelectionType Enum</td>
<td>Specifies the selection notification types.</td>
</tr>
<tr>
<td>VdSense Enum</td>
<td>Specifies the sense for labels and pins.</td>
</tr>
<tr>
<td>VdSheetSize Enum</td>
<td>Specifies sheet size.</td>
</tr>
<tr>
<td>VdSide Enum</td>
<td>Specifies the pin side.</td>
</tr>
<tr>
<td>VdSilentMode Enum</td>
<td>Specifies silent mode severity settings.</td>
</tr>
<tr>
<td>VdSourceDocumentType Enum</td>
<td>Specifies source document types.</td>
</tr>
<tr>
<td>VdSplineOrder Enum</td>
<td>Specifies spline order.</td>
</tr>
<tr>
<td>VdSplineType Enum</td>
<td>Specifies spline type.</td>
</tr>
<tr>
<td>VdSymbolType Enum</td>
<td>Specifies symbol block type.</td>
</tr>
<tr>
<td>VdTextFlags Enum</td>
<td>Specifies text flags.</td>
</tr>
<tr>
<td>VdUpdateOOScope Enum</td>
<td>Sets the scope of an update.</td>
</tr>
<tr>
<td>Enumerated Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>VdUpdateOtherObjects Enum</td>
<td>Determines which objects are updated.</td>
</tr>
<tr>
<td>VdVisibilityFlag Enum</td>
<td>Specifies attribute visibility flags.</td>
</tr>
<tr>
<td>VdWhichJoint Enum</td>
<td>Specifies joint end points.</td>
</tr>
</tbody>
</table>
PinMappingType Enum

Scope: Xpedition Designer schematic editor
Prerequisites: None
Specifies a pin mapping type.

Usage

PinMappingType. Constant

Arguments

- PinMT_ByByName
  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.\textit{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.
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

\texttt{VdAnnoPos.Constant}

Arguments

- \texttt{AnnoPosCENTER}
  Annotate at the center. The numerical value for this constant is 4.
- \texttt{AnnoPosLOWERLEFT}
  Annotate to the lower left. The numerical value for this constant is 0.
- \texttt{AnnoPosLOWERRIGHT}
  Annotate to the lower right. The numerical value for this constant is 1.
- \texttt{AnnoPosUPPERLEFT}
  Annotate to the upper left. The numerical value for this constant is 2.
- \texttt{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 124.
- 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 125.
- 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 126.
- 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 127.
- 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 128.
- 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 129.
- 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 126.

- **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 126.

- **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 131.

- **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 132.

- **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 134.

- **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 135.

- **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 136.

- **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 137.
Xpedition Designer Schematic Editor Enumerated Types

**VdAppEventDispatchID Enum**

- **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 138.

- **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 139.

- **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 140.

- **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 141.

- **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 142.

- **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 143.

- **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 144.

- **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 146.

- **AppEventSHUTDOWN**
  The event occurs when Xpedition Designer shuts down. The numerical value for this constant is 2. See “Shutdown Event (Application Object)” on page 147.

- **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 148.
• 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 149.

• AppEventSTARTUP
  The event occurs when Xpedition Designer starts. The numerical value for this constant is 1. See “Startup Event (Application Object)” on page 150.

• 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 151.

• 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 152.
# VdArcPoint Enum

Scope: Xpedition Designer schematic editor  
Prerequisites: None.  
Specifies a point that defines an arc.

## Usage

\[ \text{VdArcPoint}.\text{Constant} \]

## 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

\texttt{VdArrowType} \textit{Constant}

Arguments

- \texttt{VGARROWNONE}
  No arrowhead is drawn. The numerical value for this constant is 3.

- \texttt{VGARROWPOINT1}
  An arrowhead is drawn at the first point defining the line. The numerical value for this constant is 0.

- \texttt{VGARROWPOINT12}
  Arrowheads are drawn at both points that define the line. The numerical value for this constant is 2.

- \texttt{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.
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.
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_SCHEMA
  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.
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.
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_BUSSPIN**
  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.
Xpedition Designer Schematic Editor Enumerated Types

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.Const

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.
VdObjectType Enum

- **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

\texttt{VdOpenMode}.\texttt{Constant}

Arguments

- \texttt{VDM\_READ\_LOCK}
  Read-only (locked). The numerical value for this constant is 1.
- \texttt{VDM\_READ\_ONLY}
  Read-only (no lock). The numerical value for this constant is 2.
- \texttt{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

\textbf{VdOrientation} Constant

Arguments

- \texttt{VDORIENT.IDENTITY}
  Identity matrix index. The numerical value for this constant is 0.
- \texttt{VDORIENT.MX}
  Mirror on the x-axis (MX). The numerical value for this constant is 4.
- \texttt{VDORIENT.MX180}
  180 degree rotation on the x-axis. The numerical value for this constant is 6.
- \texttt{VDORIENT.MX270}
  270 degree rotation on the x-axis. The numerical value for this constant is 7.
- \texttt{VDORIENT.MX90}
  90 degree rotation on the x-axis. The numerical value for this constant is 5.
- \texttt{VDORIENT.ROT180}
  180 degree rotation index. The numerical value for this constant is 2.
- \texttt{VDORIENT.ROT270}
  270 degree rotation index. The numerical value for this constant is 3.
- \texttt{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).
Xpedition Designer Schematic Editor Enumerated Types

**VdParamMode Enum**

- **VDMD_COMPDEF_ON**
  The numerical value for this constant is 16 (&H10).
- **VDMD_COMPTTEXT_ON**
  Enable component text mode. The numerical value for this constant is 4.
- **VDMDCONSTRAINT**
  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).
- **VDMDDETAIL**
  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).
- **VDMDEURODATE_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).
- **VDMDFULL_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).
Xpedition Designer Schematic Editor Enumerated Types

VdParamMode Enum

- 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.
Xpedition Designer Schematic Editor Enumerated Types

VdParamValue Enum

- **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_MRÚ_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.
VdPinEndType Enum

Scope: Xpedition Designer schematic editor
Prerequisites: None.
Specifies a pin end type.

Usage

\[ \text{VdPinEndType} \ . \text{Constant} \]

Arguments

- \text{VDPIN\_BOUNDARY} 
  Boundary location (closest to bounding box). The numerical value for this constant is 1.
- \text{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.
VdSheetSize Enum

- 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.
- VDTDOP
  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.Request
```

**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

**VdUpdateOOScopeConstant**

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

VdUpdateOtherObjectsConstant

Arguments

- **VDUOO_BUSRIPPERS**
  The numerical value for this constant is 4.
- **VDUOO_BUSSIGNALS**
  The numerical value for this constant is 2.
- **VDUOO_CROSS_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.
Xpedition Designer Schematic Editor Enumerated Types

VdWhichJoint Enum
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.

![DataBook Object Model Diagram]

**Figure 6-1. DataBook Object Model**

Note

DataBook scripting does not support the C++ programming language or dual interfaces for an object.

DataBook Objects ........................................................................................................ 694
Configuring the DataBook Script ............................................................................... 722
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>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute Object</td>
<td>You can add an Attribute to the DataBook object name.</td>
</tr>
<tr>
<td>Attributes Object</td>
<td>You can add a collection of Attributes to the DataBook object name.</td>
</tr>
<tr>
<td>Component Object</td>
<td>You can add a Component to the DataBook object name.</td>
</tr>
<tr>
<td>Components Object</td>
<td>You can add a collection of Components to a DataBook object name. You can assign Count and Item properties to the Components object.</td>
</tr>
<tr>
<td>Property Object</td>
<td>You can add a Property to the DataBook object name. The Property object is a collection of properties.</td>
</tr>
</tbody>
</table>
Attribute Object

You can add an Attribute to the DataBook object name.

Table 6-2. Attribute Object Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name Property (Attribute Object)</td>
<td>Sets or returns the name of the attribute.</td>
</tr>
<tr>
<td>NameVisible Property (Attribute Object)</td>
<td>Sets or returns the visibility of the attribute name.</td>
</tr>
<tr>
<td>Value Property (Attribute Object)</td>
<td>Sets or returns the value of the attribute.</td>
</tr>
<tr>
<td>ValueVisible Property (Attribute Object)</td>
<td>Sets or returns the visibility of the attribute value.</td>
</tr>
</tbody>
</table>
Scripting with DataBook

Attribute Object

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 6-3. Attributes Object Methods and Properties

<table>
<thead>
<tr>
<th>Method or Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Method (Attributes Object)</td>
<td>Adds an attribute to the Attributes object.</td>
</tr>
<tr>
<td>Count Property (Attributes Object)</td>
<td>Returns the number of attribute objects in the Attributes object.</td>
</tr>
</tbody>
</table>
Add Method (Attributes Object)

Object: Attributes Object
Prerequisites: None
Adds an attribute to the Attributes object.

Usage

Attributes.Add (ByVal Name As String, ByVal Value As String, ByVal NameVisible As Boolean, ByVal ValueVisible 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.

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 6-4. Component Object Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes Property</td>
<td>Returns the collection of attributes for the component.</td>
</tr>
<tr>
<td>(Component Object)</td>
<td></td>
</tr>
<tr>
<td>Instances Property</td>
<td>Returns all instance values on the component as an encoded string.</td>
</tr>
<tr>
<td>(Component Object)</td>
<td></td>
</tr>
<tr>
<td>Library Property</td>
<td>Returns the library that is used in the DataBook search for the component.</td>
</tr>
<tr>
<td>(Component Object)</td>
<td></td>
</tr>
<tr>
<td>Properties Property</td>
<td>Returns a collection of one or more properties that are annotated to the</td>
</tr>
<tr>
<td>(Component Object)</td>
<td>DataBook database object.</td>
</tr>
<tr>
<td>Symbol Property</td>
<td>Sets or returns the user-defined symbol name.</td>
</tr>
<tr>
<td>(Component Object)</td>
<td></td>
</tr>
</tbody>
</table>
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

tComponent.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
attrs = 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 6-5. Components Object Methods, Properties, and Events

<table>
<thead>
<tr>
<th>Method, Property, or Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Count Property</strong> (Components Object)</td>
<td>Returns the number of component objects in the Components object.</td>
</tr>
<tr>
<td><strong>Item Method</strong> (Components Object)</td>
<td>Returns a Component object contained in the collection.</td>
</tr>
<tr>
<td><strong>Application_AddComponent Event</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Application_AfterAddComponent Event</strong></td>
<td>The Application_AfterAddComponent event is used in a DataBook script to make direct changes to the component on a schematic.</td>
</tr>
<tr>
<td><strong>Application_AfterAnnotateComponent Event</strong></td>
<td>The Application_AfterAnnotateComponent event is used in a DataBook script to make direct changes to the component on a schematic.</td>
</tr>
<tr>
<td><strong>Application_AnnotateComponent Event</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Application_LoadComponent Event</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>
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_AddGetComponent Event

Scope: Schematic editor
Object: Components Object
Prerequisites: None

You can use the Application_AddGetComponent 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_AddGetComponent (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_AddGetComponent (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
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 useful to know how for other operations
    view.SelectByName name
    MsgBox "InstanceName = " & name,"UID"
  Next
End Function
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 "pdbuslot 3 No"
MsgBox "Reset slot for " & name,"Change Slot Message"
Next
End Function
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:

```vba
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
```
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
Property Object

You can add a Property to the DataBook object name. The Property object is a collection of properties.

<table>
<thead>
<tr>
<th>Method or Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item Method (Property Object)</td>
<td>Returns a Property contained in the Property object.</td>
</tr>
<tr>
<td>Remove Method (Property Object)</td>
<td>Removes a property from the collection, using an index.</td>
</tr>
<tr>
<td>Application_SelectComponent Event</td>
<td>The Application_SelectComponent event is used in a DataBook script to update a third-party viewer when selecting different components.</td>
</tr>
<tr>
<td>Application_ViewDocument Event</td>
<td>The Application_ViewDocument event is used in a DataBook script to dynamically build URLs to web pages.</td>
</tr>
</tbody>
</table>
**Item Method (Property Object)**

Object: **Property Object**  
Prerequisites: None  
Returns a Property contained in the Property object.

**Usage**

\[ Property.Item \text{(ByVal arg As Long), or} \]

\[ Property \text{(arg)} \]

**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

\texttt{Property.Remove (ByVal arg As Long)}

Arguments

- \texttt{arg}
  
  A zero-based index of a property, or the name of a property.

Description

Applies to the Properties object (see Property Object).

Examples

\texttt{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:

```vba
Function Application_SelectComponent(Attributes)
    Dim sAttData
    For Each attr In Attributes
        attrName = attr.Name
        attrValue = attr.Value
        sAttData = sAttData & vbCrLf & attrName & "=" & attrValue
    Next
    MsgBox "The selected properties are:" & vbCrLf & sAttData,"Summary"
End Function
```
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
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 Xpedition Designer 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 ................................................................. 724
Changes to Enumerated Types .................................................. 734
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** .................................................. 724
**Changes to the Application Object** .................................. 724
**Changes to the Block Object** ....................................... 728
**Changes to the Arc Object** ........................................ 728
**Changes to the Attribute Object** ................................... 729
**Changes to the Box Object** ........................................ 729
**Changes to the Circle Object** ...................................... 730
**Changes to the Component Object** .................................. 730
**Changes to the Connection Object** .................................. 730
**Changes to the Label Object** ....................................... 731
**Changes to the Line Object** ........................................ 731
**Changes to the Net Object** ......................................... 732
**Changes to the Pin Object** .......................................... 732
**Changes to the Ripper Object** ...................................... 732
**Changes to the Text Object** ........................................ 733
**Changes to the Viewport Object** ................................... 733

**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 A-1 have been removed from the Application Object.

Note

If any of these methods or properties appear in a script, they will be ignored.

Table A-1. Changes to the Application Object

<table>
<thead>
<tr>
<th>Removed method or property</th>
<th>Reason for removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddLibrary method</td>
<td>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 488.</td>
</tr>
<tr>
<td>AddLibraryAndSaveIni method</td>
<td>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 488.</td>
</tr>
<tr>
<td>AfterDocumentSave event</td>
<td>Irrelevant now that schematics are immediately saved to the database.</td>
</tr>
<tr>
<td>AfterSaveAndCheck event</td>
<td>Irrelevant now that schematics are immediately saved to the database.</td>
</tr>
<tr>
<td>AfterSheetRead event</td>
<td>Irrelevant now that schematics are immediately saved to the database.</td>
</tr>
<tr>
<td>AfterSheetReRead event</td>
<td>Irrelevant now that schematics are immediately saved to the database.</td>
</tr>
<tr>
<td>AttributeCanHaveOatValue method</td>
<td>Instance values are always enabled in Xpedition Designer. Therefore, this method will always return a value of “True.”</td>
</tr>
<tr>
<td>AttributeValueMustBeUpper method</td>
<td>Properties in Xpedition Designer are now case-preserving and case-insensitive. Therefor, this method will always return a value of “False.”</td>
</tr>
<tr>
<td>BeforeDocumentSave event</td>
<td>Irrelevant now that schematics are immediately saved to the database.</td>
</tr>
<tr>
<td>BeforeProjectModified event</td>
<td>Replaced by BeforeProjectChanged Event (Application Object).</td>
</tr>
<tr>
<td>BeforeSaveAndCheck event</td>
<td>Irrelevant now that schematics are immediately saved to the database.</td>
</tr>
<tr>
<td>ClearAllLibraries method</td>
<td>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 333.</td>
</tr>
<tr>
<td>CnsFileString property</td>
<td>Constraints are now kept in iCDB database.</td>
</tr>
</tbody>
</table>
## Table A-1. Changes to the Application Object (cont.)

<table>
<thead>
<tr>
<th>Removed method or property</th>
<th>Reason for removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConstraintsModeChanged event</td>
<td>Constraints are now kept in iCDB database.</td>
</tr>
<tr>
<td>CurrentProject property</td>
<td>Irrelevant with the changes made to project management. You can now retrieve project data using the “GetProjectData Method (Application Object)” on page 83.</td>
</tr>
<tr>
<td>DeleteAttribute method</td>
<td>Irrelevant with the changes made to property assignment.</td>
</tr>
<tr>
<td>DeleteLibrary method</td>
<td>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 333.</td>
</tr>
<tr>
<td>DeleteLibraryAndSaveIni method</td>
<td>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 333.</td>
</tr>
<tr>
<td>DeletePackagedAttribute method</td>
<td>Irrelevant with the changes made to property assignment.</td>
</tr>
<tr>
<td>DeleteSheet method</td>
<td>Sheets are no longer identified by number; this method has been replaced by DeleteSheet Method (SchematicSheetDocuments Collection).</td>
</tr>
<tr>
<td>DocumentSave method</td>
<td>Irrelevant now that schematics are immediately saved to the database.</td>
</tr>
<tr>
<td>DocumentSaveAs method</td>
<td>Irrelevant now that schematics are immediately saved to the database.</td>
</tr>
<tr>
<td>GetAttributeValues method</td>
<td>Irrelevant with the changes made to property assignment.</td>
</tr>
<tr>
<td>GetLibraries method</td>
<td>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 333.</td>
</tr>
<tr>
<td>GetPackagedAttributeValues method</td>
<td>Irrelevant with the changes made to property assignment.</td>
</tr>
<tr>
<td>IsLibDeleteable method</td>
<td>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 333.</td>
</tr>
<tr>
<td>KickViewBase method</td>
<td>Irrelevant with removal of .wir files.</td>
</tr>
</tbody>
</table>
Changes to Xpedition Designer Automation

### Table A-1. Changes to the Application Object (cont.)

<table>
<thead>
<tr>
<th>Removed method or property</th>
<th>Reason for removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoveLibrary method</td>
<td>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 333.</td>
</tr>
<tr>
<td>MoveLibraryAndSaveIni method</td>
<td>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 333.</td>
</tr>
<tr>
<td>OATAdded event</td>
<td>Obsolete. With the changes to the database, there is no distinction specific to OAT attributes.</td>
</tr>
<tr>
<td>ObjectColor property</td>
<td>No longer used to determine the color of a schematic object.</td>
</tr>
<tr>
<td>ParamAddListName method</td>
<td>Xpedition Designer no longer has functionality to control promotions of this kind.</td>
</tr>
<tr>
<td>ParamGetListNames method</td>
<td>Xpedition Designer no longer has functionality to control promotions of this kind.</td>
</tr>
<tr>
<td>PrimaryDirectory property</td>
<td>Obsolete. You can now derive the same information using “GetProjectData Method (Application Object)” on page 83.</td>
</tr>
<tr>
<td>Project property</td>
<td>Irrelevant with the changes made to project management.</td>
</tr>
<tr>
<td>ProjectModified event</td>
<td>Replaced by ProjectChanged Event (Application Object).</td>
</tr>
<tr>
<td>Projman property</td>
<td>Irrelevant with the changes made to project management.</td>
</tr>
<tr>
<td>ReadIni method</td>
<td>Irrelevant now that Xpedition Designer now determines configuration from files other than the .ini file.</td>
</tr>
<tr>
<td>RemoveProjectSettingTab method</td>
<td>Irrelevant with the changes made to project management.</td>
</tr>
<tr>
<td>SaveIni method</td>
<td>Irrelevant since Xpedition Designer now saves all configuration files immediately upon change.</td>
</tr>
<tr>
<td>SetAttributeValues method</td>
<td>Irrelevant with the changes made to property assignment.</td>
</tr>
<tr>
<td>SetEnvVariable method</td>
<td>Obsolete</td>
</tr>
<tr>
<td>SetPackagedAttributeValue method</td>
<td>Irrelevant with the changes made to property assignment.</td>
</tr>
<tr>
<td>SynchronizesViewBase method</td>
<td>Irrelevant with removal of .wir files.</td>
</tr>
</tbody>
</table>
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 A-2 have been removed from the Block Object.

**Note**

If any of these methods or properties appear in a script, they will be ignored.

<table>
<thead>
<tr>
<th>Removed method or property</th>
<th>Reason for removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddComponent method</td>
<td>Obsolete.</td>
</tr>
<tr>
<td>AddComponentMoveMode method</td>
<td>Obsolete.</td>
</tr>
<tr>
<td>AddComponentMoveModeEx method</td>
<td>Obsolete.</td>
</tr>
<tr>
<td>AddPin method</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
<tr>
<td>AddPinAtLocation method</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
<tr>
<td>Attributes property</td>
<td>Irrelevant with the changes made to property assignment.</td>
</tr>
<tr>
<td>GetPackagedName method</td>
<td>Obsolete.</td>
</tr>
<tr>
<td>LibraryName property</td>
<td>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 333.</td>
</tr>
</tbody>
</table>

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 A-3 have been removed from the Arc Object.
Changes to Xpedition Designer Automation

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 A-4 have been removed from the Attribute Object.

Table A-4. Changes to the Attribute Object

<table>
<thead>
<tr>
<th>Removed method or property</th>
<th>Reason for removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color property</td>
<td>Irrelevant with the removal of the VdColor enumerated type.</td>
</tr>
</tbody>
</table>

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 A-5 have been removed from the Box Object.

Table A-5. Changes to the Box Object

<table>
<thead>
<tr>
<th>Removed method or property</th>
<th>Reason for removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color property</td>
<td>Irrelevant with the removal of the VdColor enumerated type.</td>
</tr>
</tbody>
</table>
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 A-6 have been removed from the Circle Object.

Note

If any of these methods or properties appear in a script, they will be ignored.

<table>
<thead>
<tr>
<th>Removed method or property</th>
<th>Reason for removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color property</td>
<td>Irrelevant with the removal of the VdColor enumerated type.</td>
</tr>
</tbody>
</table>

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 A-7 have been removed from the Component Object.

Note

If any of these methods or properties appear in a script, they will be ignored.

<table>
<thead>
<tr>
<th>Removed method or property</th>
<th>Reason for removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddBatchOats method</td>
<td>Obsolete. With the changes to the database, there is no distinction specific to OAT attributes.</td>
</tr>
</tbody>
</table>

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 A-8 have been removed from the Component Object.

Note

If any of these methods or properties appear in a script, they will be ignored.

<table>
<thead>
<tr>
<th>Removed method or property</th>
<th>Reason for removal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 A-9 have been removed from the Label Object.

**Note**

If any of these methods or properties appear in a script, they will be ignored.

<table>
<thead>
<tr>
<th>Removed method or property</th>
<th>Reason for removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color property</td>
<td>Irrelevant with the removal of the VdColor enumerated type.</td>
</tr>
<tr>
<td>Scope property</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
</tbody>
</table>

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 A-10 have been removed from the Line Object.

**Note**

If any of these methods or properties appear in a script, they will be ignored.

<table>
<thead>
<tr>
<th>Removed method or property</th>
<th>Reason for removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color property</td>
<td>Irrelevant with the removal of the VdColor enumerated type.</td>
</tr>
<tr>
<td>Scope property</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
</tbody>
</table>
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 A-11 have been removed from the Net Object.

Note
If any of these methods or properties appear in a script, they will be ignored.

<table>
<thead>
<tr>
<th>Removed method or property</th>
<th>Reason for removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddOat method</td>
<td>Obsolete. With the changes to the database, there is no distinction specific to OAT attributes.</td>
</tr>
<tr>
<td>Color property</td>
<td>Irrelevant with the removal of the VdColor enumerated type.</td>
</tr>
</tbody>
</table>

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 A-12 have been removed from the Pin Object.

Note
If any of these methods or properties appear in a script, they will be ignored.

<table>
<thead>
<tr>
<th>Removed method or property</th>
<th>Reason for removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddAttribute method</td>
<td>Obsolete due to changes in the database.</td>
</tr>
<tr>
<td>Color property</td>
<td>Obsolete due to changes in the database.</td>
</tr>
</tbody>
</table>

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 A-13 have been removed from the Ripper Object.
Note

If any of these methods or properties appear in a script, they will be ignored.

<table>
<thead>
<tr>
<th>Removed method or property</th>
<th>Reason for removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetConnectedObject method</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
<tr>
<td>GetConnectedObjects method</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
</tbody>
</table>

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 A-14 have been removed from the Text Object.

Note

If any of these methods or properties appear in a script, they will be ignored.

<table>
<thead>
<tr>
<th>Removed method or property</th>
<th>Reason for removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color property</td>
<td>Obsolete due to changes in the database.</td>
</tr>
</tbody>
</table>

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 A-15 have been removed from the Viewport Object.

Note

If any of these methods or properties appear in a script, they will be ignored.

<table>
<thead>
<tr>
<th>Removed method or property</th>
<th>Reason for removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color property</td>
<td>Obsolete due to changes in the database.</td>
</tr>
</tbody>
</table>
Changes to Xpedition Designer Automation

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** ................................................................. 734

Changes to the VdAppEventDispatchID Enumerated Type ....................... 734
Changes to the VdDocumentAccess Enumerated Type ............................... 735
Changes to the VdNotifyFlag Enumerated Type ..................................... 736
Changes to the VdObjectClass Enumerated Type .................................. 737
Changes to the VdObjectType Enumerated Type ................................... 737
Changes to the VdObjectTypeMask Enumerated Type ............................. 738

**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 A-16 have been removed from the VdAppEventDispatchID Enum.
Note

If any of these appear in a script, they will be ignored.

Table A-16. Changes to the VdAppEventDispatchID Enumerated Type

<table>
<thead>
<tr>
<th>Removed constant</th>
<th>Reason for removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFTER_DOCUMENT_SAVE</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
<tr>
<td>AFTER_SAVE_AND_CHECK</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
<tr>
<td>AFTER_SHEET_REREAD</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
<tr>
<td>BEFORE_DOCUMENT_SAVE</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
<tr>
<td>BEFORE_PROJECT_MODIFIED</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
<tr>
<td>BEFORE_SAVE_AND_CHECK</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
<tr>
<td>CONSTRAINTS_MODE_CHANGED</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
<tr>
<td>DOCUMENT_SAVE</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
<tr>
<td>DOCUMENT_SAVEAS</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
<tr>
<td>OAT_ADDED</td>
<td>Obsolete. With the changes to the database, there is no distinction specific to OAT attributes.</td>
</tr>
<tr>
<td>PROJECT_MODIFIED</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
<tr>
<td>SOURCE_FILE_MODIFIED</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
<tr>
<td>SOURCEDOCUMENT_SAVE</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
</tbody>
</table>

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.
Changes to Xpedition Designer Automation

Changes to the VdNotifyFlag Enumerated Type

The items shown in Table A-17 have been removed from the VdDocumentAccess Enum.

---

**Note**

If any of these appear in a script, they will be ignored.

---

### Table A-17. Changes to the VdDocumentAccess Enumerated Type

<table>
<thead>
<tr>
<th>Removed constant</th>
<th>Reason for removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOCK_WRITABLE</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
<tr>
<td>OAT_READ_ONLY</td>
<td>Obsolete. With the changes to the database, there is no distinction specific to OAT attributes.</td>
</tr>
<tr>
<td>OAT_WRITABLE</td>
<td>Obsolete. With the changes to the database, there is no distinction specific to OAT attributes.</td>
</tr>
<tr>
<td>SYM_READ_LOCK</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
<tr>
<td>SYM_READ_ONLY</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
<tr>
<td>SYM_WRITABLE</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
<tr>
<td>WIR_READ_ONLY</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
<tr>
<td>WIR_WRITABLE</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
</tbody>
</table>

---

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 A-18 have been removed from the VdNotifyFlag Enum.

---

**Note**

If any of these appear in a script, they will be ignored.

---

### Table A-18. Changes to the VdNotifyFlag Enumerated Type

<table>
<thead>
<tr>
<th>Removed constant</th>
<th>Reason for removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOC_SAVE_AFTER</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
</tbody>
</table>
Changes to Xpedition Designer Automation

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 A-19 have been removed from the VdObjectClass Enum.

Note

If any of these appear in a script, they will be ignored.

<table>
<thead>
<tr>
<th>Removed constant</th>
<th>Reason for removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>OAT</td>
<td>Obsolete. Obsolete. With the changes to the database, there is no distinction specific to OAT attributes.</td>
</tr>
<tr>
<td>PIN</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
<tr>
<td>SYMBOL</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
<tr>
<td>WIRELIST</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
</tbody>
</table>

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 A-20 have been removed from the VdObjectType Enum.

Note

If any of these appear in a script, they will be ignored.

<table>
<thead>
<tr>
<th>Removed constant</th>
<th>Reason for removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOC_SAVE_BEFORE</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
<tr>
<td>DOC_SAVEAS_AFTER</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
<tr>
<td>DOC_SAVEAS_BEFORE</td>
<td>Irrelevant with the removal of .wir and .sch files and changes to the database paradigm.</td>
</tr>
</tbody>
</table>
Changes to Xpedition Designer Automation

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 A-20 have been removed from the VdObjectTypeMask Enum.

**Note**

If any of these appear in a script, they will be ignored.

<table>
<thead>
<tr>
<th>Removed constant</th>
<th>Reason for removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIBRARY</td>
<td>Irrelevant with the removal of <code>.wir</code> and <code>.sch</code> files and changes to the database paradigm.</td>
</tr>
<tr>
<td>OATATTRIBUTE</td>
<td>Obsolete. With the changes to the database, there is no distinction specific to OAT attributes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Removed constant</th>
<th>Reason for removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>OAT</td>
<td>Obsolete. With the changes to the database, there is no distinction specific to OAT attributes.</td>
</tr>
</tbody>
</table>
End-User License Agreement

The latest version of the End-User License Agreement is available on-line at: www.mentor.com/eula

IMPORTANT INFORMATION

USE OF ALL SOFTWARE IS SUBJECT TO LICENSE RESTRICTIONS. CAREFULLY READ THIS LICENSE AGREEMENT BEFORE USING THE PRODUCTS. USE OF SOFTWARE INDICATES CUSTOMER’S COMPLETE AND UNCONDITIONAL ACCEPTANCE OF THE TERMS AND CONDITIONS SET FORTH IN THIS AGREEMENT. ANY ADDITIONAL OR DIFFERENT PURCHASE ORDER TERMS AND CONDITIONS SHALL NOT APPLY.

END-USER LICENSE AGREEMENT ("Agreement")

This is a legal agreement concerning the use of Software (as defined in Section 2) and hardware (collectively “Products”) between the company acquiring the Products ("Customer"), and the Mentor Graphics entity that issued the corresponding quotation or, if no quotation was issued, the applicable local Mentor Graphics entity ("Mentor Graphics"). Except for license agreements related to the subject matter of this license agreement which are physically signed by Customer and an authorized representative of Mentor Graphics, this Agreement and the applicable quotation contain the parties' entire understanding relating to the subject matter and supersede all prior or contemporaneous agreements. If Customer does not agree to these terms and conditions, promptly return or, in the case of Software received electronically, certify destruction of Software and all accompanying items within five days after receipt of Software and receive a full refund of any license fee paid.

1. ORDERS, FEES AND PAYMENT.

1.1. To the extent Customer (or if agreed by Mentor Graphics, Customer’s appointed third party buying agent) places and Mentor Graphics accepts purchase orders pursuant to this Agreement (each an “Order”), each Order will constitute a contract between Customer and Mentor Graphics, which shall be governed solely and exclusively by the terms and conditions of this Agreement, any applicable addenda and the applicable quotation, whether or not those documents are referenced on the Order. Any additional or conflicting terms and conditions appearing on an Order or presented in any electronic portal or automated order management system, whether or not required to be electronically accepted, will not be effective unless agreed in writing and physically signed by an authorized representative of Customer and Mentor Graphics.

1.2. Amounts invoiced will be paid, in the currency specified on the applicable invoice, within 30 days from the date of such invoice. Any past due invoices will be subject to the imposition of interest charges in the amount of one and one-half percent per month or the applicable legal rate currently in effect, whichever is lower. Prices do not include freight, insurance, customs duties, taxes or other similar charges, which Mentor Graphics will state separately in the applicable invoice. Unless timely provided with a valid certificate of exemption or other evidence that items are not taxable, Mentor Graphics will invoice Customer for all applicable taxes including, but not limited to, VAT, GST, sales tax, consumption tax and service tax. Customer will make all payments free and clear of, and without reduction for, any withholding or other taxes; any such taxes imposed on payments by Customer hereunder will be Customer’s sole responsibility. If Customer appoints a third party to place purchase orders and/or make payments on Customer’s behalf, Customer shall be liable for payment under Orders placed by such third party in the event of default.

1.3. All Products are delivered FCA factory (Incoterms 2010), freight prepaid and invoiced to Customer, except Software delivered electronically, which shall be deemed delivered when made available to Customer for download. Mentor Graphics retains a security interest in all Products delivered under this Agreement, to secure payment of the purchase price of such Products, and Customer agrees to sign any documents that Mentor Graphics determines to be necessary or convenient for use in filing or perfecting such security interest. Mentor Graphics’ delivery of Software by electronic means is subject to Customer’s provision of both a primary and an alternate e-mail address.

2. GRANT OF LICENSE. The software installed, downloaded, or otherwise acquired by Customer under this Agreement, including any updates, modifications, revisions, copies, documentation, setup files and design data ("Software") are copyrighted, trade secret and confidential information of Mentor Graphics or its licensors, who maintain exclusive title to all Software and retain all rights not expressly granted by this Agreement. Except for Software that is embeddable ("Embedded Software"), which is licensed pursuant to separate embedded software terms or an embedded software supplement, Mentor Graphics grants to Customer, subject to payment of applicable license fees, a nontransferable, nonexclusive license to use Software solely: (a) in machine-readable, object-code form (except as provided in Subsection 4.2); (b) for Customer’s internal business purposes; (c) for the term of the license; and (d) on the computer hardware and at the site authorized by Mentor Graphics. A site is restricted to a one-half mile (800 meter) radius. Customer may have Software temporarily used by an employee for telecommuting purposes from locations other than a Customer office, such as the employee’s residence, an airport or hotel, provided that such employee’s primary place of employment is the site where the Software is authorized for use. Mentor Graphics’ standard policies and programs, which vary depending on Software, license fees paid or services purchased, apply to the following: (a) relocation of Software; (b) use of Software, which may be limited, for example, to execution of a single session by a single user on the authorized hardware or for a restricted period of time (such limitations may be technically implemented through the use of authorization codes or similar devices); and (c) support services provided, including eligibility to receive telephone support, updates, modifications, and revisions. For the avoidance of doubt, if Customer provides any feedback or requests any change or enhancement to Products, whether in the course of receiving support or consulting services, evaluating Products, performing beta testing or otherwise, any inventions, product improvements, modifications or developments made by Mentor Graphics (at Mentor Graphics’ sole discretion) will be the exclusive property of Mentor Graphics.
3. BETA CODE.

3.1. Portions or all of certain Software may contain code for experimental testing and evaluation (which may be either alpha or beta, collectively “Beta Code”), which may not be used without Mentor Graphics’ explicit authorization. Upon Mentor Graphics’ authorization, Mentor Graphics grants to Customer a temporary, nontransferable, nonexclusive license for experimental use to test and evaluate the Beta Code without charge for a limited period of time specified by Mentor Graphics. Mentor Graphics may choose, at its sole discretion, not to release Beta Code commercially in any form.

3.2. If Mentor Graphics authorizes Customer to use the Beta Code, Customer agrees to evaluate and test the Beta Code under normal conditions as directed by Mentor Graphics. Customer will contact Mentor Graphics periodically during Customer’s use of the Beta Code to discuss any malfunctions or suggested improvements. Upon completion of Customer’s evaluation and testing, Customer will send to Mentor Graphics a written evaluation of the Beta Code, including its strengths, weaknesses and recommended improvements.

3.3. Customer agrees to maintain Beta Code in confidence and shall restrict access to the Beta Code, including the methods and concepts utilized therein, solely to those employees and Customer location(s) authorized by Mentor Graphics to perform beta testing. Customer agrees that any written evaluations and all inventions, product improvements, modifications or developments that Mentor Graphics conceived or made during or subsequent to this Agreement, including those based partly or wholly on Customer’s feedback, will be the exclusive property of Mentor Graphics. Mentor Graphics will have exclusive rights, title and interest in all such property. The provisions of this Subsection 3.3 shall survive termination of this Agreement.

4. RESTRICTIONS ON USE.

4.1. Customer may copy Software only as reasonably necessary to support the authorized use. Each copy must include all notices and legends embedded in Software and affixed to its medium and container as received from Mentor Graphics. All copies shall remain the property of Mentor Graphics or its licensors. Except for Embedded Software that has been embedded in executable code form in Customer’s product(s), Customer shall maintain a record of the number and primary location of all copies of Software, including copies merged with other software, and shall make those records available to Mentor Graphics upon request. Customer shall not make Products available in any form to any person other than Customer’s employees and on-site contractors, excluding Mentor Graphics competitors, whose job performance requires access and who are under obligations of confidentiality. Customer shall take appropriate action to protect the confidentiality of Products and ensure that any person permitted access does not disclose or use Products except as permitted by this Agreement. Customer shall give Mentor Graphics written notice of any unauthorized disclosure or use of the Products as soon as Customer becomes aware of such unauthorized disclosure or use. Customer acknowledges that Software provided hereunder may contain source code which is proprietary and its confidentiality is of the highest importance and value to Mentor Graphics. Customer acknowledges that Mentor Graphics may be seriously harmed if such source code is disclosed in violation of this Agreement. Except as otherwise permitted for purposes of interoperability as specified by applicable and mandatory local law, Customer shall not reverse-assemble, disassemble, reverse-compile, or reverse-engineer any Product, or in any way derive any source code from Software that is not provided to Customer in source code form. Log files, data files, rule files and script files generated by or for the Software (collectively “Files”), including without limitation files containing Standard Verification Rule Format (“SVRF”) and Tcl Verification Format (“TVF”) which are Mentor Graphics’ trade secret and proprietary syntaxes for expressing process rules, constitute or include confidential information of Mentor Graphics. Customer may share Files with third parties, excluding Mentor Graphics competitors, provided that the confidentiality of such Files is protected by written agreement at least as well as Customer protects other information of a similar nature or importance, but in any case with at least reasonable care. Customer may use Files containing SVRF or TVF only with Mentor Graphics products. Under no circumstances shall Customer use Products or Files or allow their use for the purpose of developing, enhancing or marketing any product that in any way competitive with Products, or disclose to any third party the results of, or information pertaining to, any benchmark.

4.2. If any Software or portions thereof are provided in source code form, Customer will use the source code only to correct software errors and enhance or modify the Software for the authorized use, or as permitted for Embedded Software under separate embedded software terms or an embedded software supplement. Customer shall not disclose or permit disclosure of source code, in whole or in part, including any of its methods or concepts, to anyone except Customer’s employees or on-site contractors, excluding Mentor Graphics competitors, with a need to know. Customer shall not copy or compile source code in any manner except to support this authorized use.

4.3. Customer agrees that it will not subject any Product to any open source software (“OSS”) license that conflicts with this Agreement or that does not otherwise apply to such Product.

4.4. Customer may not assign this Agreement or the rights and duties under it, or relocate, sublicense, or otherwise transfer the Products, whether by operation of law or otherwise (“Attempted Transfer”), without Mentor Graphics’ prior written consent and payment of Mentor Graphics’ then-current applicable relocation and/or transfer fees. Any Attempted Transfer without Mentor Graphics’ prior written consent shall be a material breach of this Agreement and may, at Mentor Graphics’ option, result in the immediate termination of the Agreement and/or the licenses granted under this Agreement. The terms of this Agreement, including without limitation the licensing and assignment provisions, shall be binding upon Customer’s permitted successors in interest and assigns.

4.5. The provisions of this Section 4 shall survive the termination of this Agreement.

5. SUPPORT SERVICES. To the extent Customer purchases support services, Mentor Graphics will provide Customer with updates and technical support for the Products, at the Customer site(s) for which support is purchased, in accordance with Mentor Graphics’ then current End-User Support Terms located at http://supportnet.mentor.com/supportterms.

6. OPEN SOURCE SOFTWARE. Products may contain OSS or code distributed under a proprietary third party license agreement, to which additional rights or obligations (“Third Party Terms”) may apply. Please see the applicable Product documentation (including license files, header files, read-me files or source code) for details. In the event of conflict between the terms of this Agreement
7. **LIMITED WARRANTY.**

7.1. Mentor Graphics warrants that during the warranty period its standard, generally supported Products, when properly installed, will substantially conform to the functional specifications set forth in the applicable user manual. Mentor Graphics does not warrant that Products will meet Customer’s requirements or that operation of Products will be uninterrupted or error free. The warranty period is 90 days starting on the 15th day after delivery or upon installation, whichever first occurs. Customer must notify Mentor Graphics in writing of any nonconformity within the warranty period. For the avoidance of doubt, this warranty applies only to the initial shipment of Software under an Order and does not renew or reset, for example, with the delivery of (a) Software updates or (b) authorization codes or alternate Software under a transaction involving Software re-mix. This warranty shall not be valid if Products have been subject to misuse, unauthorized modification, improper installation or Customer is not in compliance with this Agreement. MENTOR GRAPHICS’ ENTIRE LIABILITY AND CUSTOMER’S EXCLUSIVE REMEDY SHALL BE, AT MENTOR GRAPHICS’ OPTION, EITHER (A) REFUND OF THE PRICE PAID UPON RETURN OF THE PRODUCTS TO MENTOR GRAPHICS OR (B) MODIFICATION OR REPLACEMENT OF THE PRODUCTS THAT DO NOT MEET THIS LIMITED WARRANTY. MENTOR GRAPHICS MAKES NO WARRANTIES WITH RESPECT TO: (A) SERVICES; (B) PRODUCTS PROVIDED AT NO CHARGE; OR (C) BETA CODE; ALL OF WHICH ARE PROVIDED “AS IS.”

7.2. THE WARRANTIES SET FORTH IN THIS SECTION 7 ARE EXCLUSIVE. NEITHER MENTOR GRAPHICS NOR ITS LICENSORS MAKE ANY OTHER WARRANTIES EXPRESS, IMPLIED OR STATUTORY, WITH RESPECT TO PRODUCTS PROVIDED UNDER THIS AGREEMENT. MENTOR GRAPHICS AND ITS LICENSORS SPECIFICALLY DISCLAIM ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY.

8. **LIMITATION OF LIABILITY.** TO THE EXTENT PERMITTED UNDER APPLICABLE LAW, IN NO EVENT SHALL MENTOR GRAPHICS OR ITS LICENSORS BE LIABLE FOR INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES (INCLUDING LOST PROFITS OR SAVINGS) WHETHER BASED ON CONTRACT, TORT OR ANY OTHER LEGAL THEORY, EVEN IF MENTOR GRAPHICS OR ITS LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT SHALL MENTOR GRAPHICS’ OR ITS LICENSORS’ LIABILITY UNDER THIS AGREEMENT EXCEED THE AMOUNT RECEIVED FROM CUSTOMER FOR THE HARDWARE, SOFTWARE LICENSE OR SERVICE GIVING RISE TO THE CLAIM. IN THE CASE WHERE NO AMOUNT WAS PAID, MENTOR GRAPHICS AND ITS LICENSORS SHALL HAVE NO LIABILITY FOR ANY DAMAGES WHATSOEVER. THE PROVISIONS OF THIS SECTION 8 SHALL SURVIVE THE TERMINATION OF THIS AGREEMENT.

9. **THIRD PARTY CLAIMS.**

9.1. Customer acknowledges that Mentor Graphics has no control over the testing of Customer’s products, or the specific applications and use of Products. Mentor Graphics and its licensors shall not be liable for any claim or demand made against Customer by any third party, except to the extent such claim is covered under Section 10.

9.2. In the event that a third party makes a claim against Mentor Graphics arising out of the use of Customer’s products, Mentor Graphics will give Customer prompt notice of such claim. At Customer’s option and expense, Customer may take sole control of the defense and any settlement of such claim. Customer WILL reimburse and hold harmless Mentor Graphics for any LIABILITY, damages, settlement amounts, costs and expenses, including reasonable attorney’s fees, incurred by or awarded against Mentor Graphics or its licensors in connection with such claims.

9.3. The provisions of this Section 9 shall survive any expiration or termination of this Agreement.

10. **INFRINGEMENT.**

10.1. Mentor Graphics will defend or settle, at its option and expense, any action brought against Customer in the United States, Canada, Japan, or member state of the European Union which alleges that any standard, generally supported Product acquired by Customer hereunder infringes a patent or copyright or misappropriates a trade secret in such jurisdiction. Mentor Graphics will pay costs and damages finally awarded against Customer that are attributable to such action. Customer understands and agrees that as conditions to Mentor Graphics’ obligations under this section Customer must: (a) notify Mentor Graphics promptly in writing of the action; (b) provide Mentor Graphics all reasonable information and assistance to settle or defend the action; and (c) grant Mentor Graphics sole authority and control of the defense or settlement of the action.

10.2. If a claim is made under Subsection 10.1 Mentor Graphics may, at its option and expense: (a) replace or modify the Product so that it becomes noninfringing; (b) procure for Customer the right to continue using the Product; or (c) require the return of the Product and refund to Customer any purchase price or license fee paid, less a reasonable allowance for use.

10.3. Mentor Graphics has no liability to Customer if the action is based upon: (a) the combination of Software or hardware with any product not furnished by Mentor Graphics; (b) the modification of the Product other than by Mentor Graphics; (c) the use of other (including lost profits or savings) whether based on contract, tort or any other legal theory, even if Mentor Graphics or its licensors have been advised of the possibility of such damages. In no event shall Mentor Graphics’ or its licensors’ liability under this agreement exceed the amount received from customer for the hardware, software license or service giving rise to the claim. In the case where no amount was paid, Mentor Graphics and its licensors shall have no liability for any damages whatsoever. The provisions of this Section 8 shall survive the termination of this agreement.

10.4. THIS SECTION 10 IS SUBJECT TO SECTION 8 ABOVE AND STATES THE ENTIRE LIABILITY OF MENTOR GRAPHICS AND ITS LICENSORS, AND CUSTOMER’S SOLE AND EXCLUSIVE REMEDY, FOR DEFENSE,
11. TERMINATION AND EFFECT OF TERMINATION.

11.1. If a Software license was provided for limited term use, such license will automatically terminate at the end of the authorized term. Mentor Graphics may terminate this Agreement and/or any license granted under this Agreement immediately upon written notice if Customer: (a) exceeds the scope of the license or otherwise fails to comply with the licensing or confidentiality provisions of this Agreement, or (b) becomes insolvent, files a bankruptcy petition, institutes proceedings for liquidation or winding up or enters into an agreement to assign its assets for the benefit of creditors. For any other material breach of any provision of this Agreement, Mentor Graphics may terminate this Agreement and/or any license granted under this Agreement upon 30 days written notice if Customer fails to cure the breach within the 30 day notice period. Termination of this Agreement or any license granted hereunder will not affect Customer’s obligation to pay for Products shipped or licenses granted prior to the termination, which amounts shall be payable immediately upon the date of termination.

11.2. Upon termination of this Agreement, the rights and obligations of the parties shall cease except as expressly set forth in this Agreement. Upon termination of this Agreement and/or any license granted under this Agreement, Customer shall ensure that all use of the affected Products ceases, and shall return hardware and either return to Mentor Graphics or destroy Software in Customer’s possession, including all copies and documentation, and certify in writing to Mentor Graphics within ten business days of the termination date that Customer no longer possesses any of the affected Products or copies of Software in any form.

12. EXPORT. The Products provided hereunder are subject to regulation by local laws and European Union (“E.U.”) and United States (“U.S.”) government agencies, which prohibit export, re-export or diversion of certain products, information about the products, and direct or indirect products thereof, to certain countries and certain persons. Customer agrees that it will not export or re-export Products in any manner without first obtaining all necessary approval from appropriate local, E.U. and U.S. government agencies. If Customer wishes to disclose any information to Mentor Graphics that is subject to any E.U., U.S. or other applicable export restrictions, including without limitation the U.S. International Traffic in Arms Regulations (ITAR) or special controls under the Export Administration Regulations (EAR), Customer will notify Mentor Graphics personnel, in advance of each instance of disclosure, that such information is subject to such export restrictions.

13. U.S. GOVERNMENT LICENSE RIGHTS. Software was developed entirely at private expense. The parties agree that all Software is commercial computer software within the meaning of the applicable acquisition regulations. Accordingly, pursuant to U.S. FAR 48 CFR 12.212 and DFAR 48 CFR 227.7202, use, duplication and disclosure of the Software by or for the U.S. government or a U.S. commercial computer software within the meaning of the applicable acquisition regulations. Accordingly, pursuant to U.S. FAR 48 CFR 12.212 and DFAR 48 CFR 227.7202, use, duplication and disclosure of the Software by or for the U.S. government or a U.S. government contractor is subject to such export restrictions.

14. THIRD PARTY BENEFICIARY. Mentor Graphics Corporation, Mentor Graphics (Ireland) Limited, Microsoft Corporation and other licensors may be third party beneficiaries of this Agreement with the right to enforce the obligations set forth herein.

15. REVIEW OF LICENSE USAGE. Customer will monitor the access to and use of Software. With prior written notice and during Customer’s normal business hours, Mentor Graphics may engage an internationally recognized accounting firm to review Customer’s software monitoring system and records deemed relevant by the internationally recognized accounting firm to confirm Customer’s compliance with the terms of this Agreement or U.S. or other local export laws. Such review may include FlexNet (or successor product) report log files that Customer shall capture and provide at Mentor Graphics’ request. Customer shall make records available in electronic format and shall fully cooperate with data gathering to support the license review. Mentor Graphics shall bear the expense of any such review unless a material non-compliance is revealed. Mentor Graphics shall treat as confidential information all information gained as a result of any request or review and shall only use or disclose such information as required by law or to enforce its rights under this Agreement. The provisions of this Section 15 shall survive the termination of this Agreement.

16. CONTROLLING LAW, JURISDICTION AND DISPUTE RESOLUTION. The owners of certain Mentor Graphics intellectual property licensed under this Agreement are located in Ireland and the U.S. To promote consistency around the world, disputes shall be resolved as follows: excluding conflict of laws rules, this Agreement shall be governed by and construed under the laws of the State of Oregon, U.S., if Customer is located in North or South America, and the laws of Ireland if Customer is located outside of North or South America or Japan, and the laws of Japan if Customer is located in Japan. All disputes arising out of or in relation to this Agreement shall be submitted to the exclusive jurisdiction of the courts of Portland, Oregon when the laws of Oregon apply, or Dublin, Ireland when the laws of Ireland apply, or the Tokyo District Court when the laws of Japan apply. Notwithstanding the foregoing, all disputes in Asia (excluding Japan) arising out of or in relation to this Agreement shall be resolved by arbitration in Singapore before a single arbitrator to be appointed by the chairman of the Singapore International Arbitration Centre (“SIAC”) to be conducted in the English language, in accordance with the Arbitration Rules of the SIAC in effect at the time of the dispute, which rules are deemed to be incorporated by reference in this section. Nothing in this section shall restrict Mentor Graphics’ right to bring an action (including for example a motion for injunctive relief) against Customer in the jurisdiction where Customer’s place of business is located. The United Nations Convention on Contracts for the International Sale of Goods does not apply to this Agreement.

17. Severability. If any provision of this Agreement is held by a court of competent jurisdiction to be void, invalid, unenforceable or illegal, such provision shall be severed from this Agreement and the remaining provisions will remain in full force and effect.

18. MISCELLANEOUS. This Agreement contains the parties’ entire understanding relating to its subject matter and supersedes all prior or contemporaneous agreements. Any translation of this Agreement is provided to comply with local legal requirements only. In the event of a dispute between the English and any non-English versions, the English version of this Agreement shall govern to the extent not prohibited by local law in the applicable jurisdiction. This Agreement may only be modified in writing, signed by an authorized representative of each party. Waiver of terms or excuse of breach must be in writing and shall not constitute subsequent consent, waiver or excuse.

Rev. 170330, Part No. 270941