

# AVEVA™ Industrial Graphic Editor User Guide



© 2020 AVEVA Group plc and its subsidiaries. All rights reserved.

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

Although precaution has been taken in the preparation of this documentation, AVEVA assumes no responsibility for errors or omissions. The information in this documentation is subject to change without notice and does not represent a commitment on the part of AVEVA. The software described in this documentation is furnished under a license agreement. This software may be used or copied only in accordance with the terms of such license agreement.

ArchestrA, Aquis, Avantis, Citect, DYNsIM, eDNA, EYESIM, InBatch, InduSoft, InStep, IntelaTrac, InTouch, OASyS, PIPEPHASE, PRiSM, PRO/II, PROVISION, ROMeo, SIM4ME, SimCentral, SimSci, Skelta, SmartGlance, Spiral Software, Termis, WindowMaker, WindowViewer, and Wonderware are trademarks of AVEVA and/or its subsidiaries. An extensive listing of AVEVA trademarks can be found at: <https://sw.aveva.com/legal>. All other brands may be trademarks of their respective owners.

Publication date: Thursday, November 19, 2020

### **Contact Information**

AVEVA Group plc  
High Cross  
Madingley Road  
Cambridge  
CB3 0HB. UK

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

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

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

# Contents

Chapter 1 About AVEVA Industrial Graphics .....	17
The Industrial Graphic Editor .....	17
Tools Panel.....	18
Elements List.....	19
Properties Editor .....	20
Animations Summary .....	20
Canvas .....	20
Elements .....	20
Basic Elements .....	21
Status Element.....	21
Windows Common Controls .....	22
Groups .....	23
Path Graphics .....	24
Windows Client Controls .....	24
Properties .....	25
Predefined Properties .....	25
Custom Properties.....	25
Properties of Groups.....	25
Animations.....	27
Animation Types.....	27
Data Sources for Animations .....	29
Animation Capabilities of Groups.....	29
Animation States .....	30
Embedding Graphics.....	31
Changing Embedded Graphics.....	32
Embedding and Instantiation .....	32
Graphic Change Propagation.....	33
Size Propagation and Anchor Points.....	34
Estimating Graphic Performance.....	35
Estimating Graphics' Performance.....	35
Understanding GPI Rating Calculations .....	37
Elements Category .....	37
Animations Category .....	38
Styles Category .....	39
Reference Category.....	40
Custom Properties Category .....	40
Scripts Category .....	41
Examining a Graphic with a 4.5 GPI Rating.....	41
Saving a Graphic that May Impact Runtime Performance.....	43
Showing Quality and Status .....	43
Showing Quality and Status with the Status Element .....	43
Showing Quality and Status by Overriding .....	44
Managing Industrial Graphics.....	45
Creating a New Graphic .....	45

Opening Graphics for Editing .....	45
Organizing Graphics .....	45
Importing and Exporting Graphics .....	45
Deleting a Graphic .....	45
Creating Multiple Configurations of a Graphic .....	46
Understanding Visual and Functional Graphic Configurations .....	46
Visual Graphic Configurations .....	46
Functional Graphic Configurations .....	46
Embedding Graphics .....	47
Appearance of Embedded Graphics .....	47
Changing Embedded Graphics .....	47
Viewing a Graphic in Read-Only Mode .....	47
<b>Using the Industrial Graphic Editor .....</b>	<b>49</b>
Showing, Hiding and Adjusting Panels .....	49
Panning and Zooming the Canvas .....	49
Panning .....	49
Using the Pan and Zoom Window to Pan .....	49
Using the Hand Tool to Pan .....	50
Using the Mouse Scroll Wheel to Pan .....	50
Zooming .....	50
Zooming In to a Specified Point .....	51
Zooming Out from a Specified Point .....	51
Zooming to the Default Zoom Value .....	51
Zooming a Selected Element .....	51
Zooming a Specified Area .....	51
Selecting or Specifying a Zoom Value .....	52
Using the Pan and Zoom Window to Change the Zoom .....	52
Using the Mouse Scroll Wheel for Zooming .....	52
Configuring Designer Preferences .....	52
Using the Symbol Wizard Editor .....	53
<b>Working with Graphic Elements .....</b>	<b>57</b>
About Graphic Elements .....	57
Drawing and Dragging Elements .....	57
Drawing Rectangles, Rounded Rectangles, Ellipses, and Lines .....	58
Drawing Polylines, Polygons, Curves, and Closed Curves .....	58
Drawing 2-Point Arcs, 2-Point Pies and 2-Point Chords .....	58
Drawing 3-Point Arcs, 3-Point Pies, and 3-Point Chords .....	58
Placing and Importing Images .....	59
Drawing Buttons .....	59
Placing Text .....	59
Drawing Text Boxes .....	59
Drawing Status Elements .....	60
Drawing User Interface Common Controls .....	60
Dragging Elements .....	60
Editing Element Properties .....	61
Selecting Elements .....	62

Selecting Elements by Mouse Click .....	63
Selecting Elements by Lasso .....	63
Selecting All Elements .....	63
Selecting Elements Using the Elements List.....	64
Unselecting Elements .....	64
Inline Editing .....	64
Copying, Cutting, and Pasting Elements .....	65
Copying Elements .....	65
Cutting or Deleting Elements .....	66
Duplicating Elements .....	66
Moving Elements .....	67
Aligning Elements .....	68
Aligning Elements Horizontally .....	68
Aligning Elements Vertically .....	69
Aligning Elements by their Center Points .....	70
Aligning Elements by their Points of Origin .....	70
Adjusting the Spacing between Elements .....	70
Distributing Elements.....	71
Making Space between Elements Equal .....	71
Increasing Space between Elements .....	71
Decreasing Space between Elements.....	72
Removing All Space between Elements .....	72
Resizing Elements .....	72
Resizing a Single Element with the Mouse .....	73
Resizing Elements by Changing Size Properties .....	73
Resizing Elements Proportionally .....	73
Making Elements the Same Width, Height, or Size .....	74
Adjusting the z-Order of Elements .....	74
Rotating Elements.....	75
Rotating Elements with the Mouse.....	76
Rotating Elements by Changing the Angle Property .....	76
Rotating Elements by 90 Degrees .....	77
Moving the Origin of an Element .....	77
Changing Points of Origin with the Mouse .....	77
Changing Points of Origin in the Properties Editor .....	77
Add Connectors Between Graphic Elements.....	78
Draw a Connector .....	79
Adding Connection Points .....	80
Change Connector Properties .....	81
Change the Type of Connector .....	82
Change the Length of a Connector .....	82
Change the Shape of a Connector .....	83
Flipping Elements .....	84
Locking and Unlocking Elements .....	85
Making Changes Using Undo and Redo .....	85
Working with Groups of Elements .....	86
Creating a Group of Elements .....	86
Ungrouping.....	87

Adding Elements to Existing Groups .....	87
Removing Elements from Groups .....	87
Editing Components within a Group .....	88
Using Path Graphics .....	88
Creating a Path Graphic .....	89
Breaking the Path of a Path Graphic .....	89
Changing a Path Graphic .....	90
Moving Elements in a Path Graphic .....	90
Resizing Elements in a Path Graphic .....	90
Editing Start and Sweep Angles of Elements in a Path Graphic .....	91
Editing Element Control Points in a Path Graphic .....	91
Swapping the End Points of an Element in a Path Graphic .....	91
Changing the Z-order of an Element in a Path Graphic .....	92
Adding Elements to an Existing Path Graphic .....	93
Removing Elements from a Path Graphic .....	94
<b>Editing Common Properties of Elements and Graphics .....</b>	<b>95</b>
Editing the Name of an Element .....	95
Editing the Fill Properties of an Element .....	95
Setting Fill Style .....	96
Setting Unfilled Style .....	97
Setting Fill Orientation .....	97
Setting Fill Behavior .....	97
Setting Horizontal Fill Direction and Percentage .....	97
Setting Vertical Fill Direction and Percentage .....	98
Editing the Line Properties of an Element .....	98
Setting Start or End Points of a Line .....	98
Setting the Line Weight .....	99
Setting the Line Pattern .....	99
Setting the Line Style .....	99
Setting the Text Properties of an Element .....	100
Setting the Displayed Text .....	100
Setting the Text Display Format .....	100
Setting the Text Font .....	100
Setting the Text Color .....	101
Setting the Text Alignment .....	101
Substituting Strings .....	102
Setting Style .....	103
Setting a Solid Color .....	103
Setting a Solid Color from the Standard Palette .....	104
Setting a Solid Color from the Color Disc and Bar .....	104
Setting a Solid Color with the Value Input Boxes .....	104
Setting a Solid Color with the Color Picker .....	105
Setting a Solid Color from the Custom Palette .....	105
Adding and Removing Colors in the Custom Palette .....	105
Saving and Loading the Custom Palette .....	106
Setting a Gradient .....	106
Setting the Number of Colors for a Gradient .....	106
Setting the Direction of the Gradient .....	107
Changing the Variant of a Gradient .....	108
Setting the Color Distribution Shape .....	108
Setting the Focus Scales of a Gradient .....	109

---

Setting a Pattern .....	110
Setting a Texture .....	110
Setting the Style to No Fill .....	111
Setting the Transparency of a Style .....	111
Setting the Transparency Level of an Element .....	111
Adjusting the Colors and Transparency of a Gradient .....	112
Loading Graphics with Deprecated Features .....	112
Enabling and Disabling Elements for Run-Time Interaction .....	112
Changing the Visibility of Elements .....	113
Editing the Tab Order of an Element .....	113
Using the Format Painter to Format Elements .....	114
Editing the General Properties of a Graphic .....	115
<b>Editing Graphic-Specific and Element-Specific Properties .....</b>	<b>117</b>
About Graphic- and Element-Specific Properties .....	117
Setting the Radius of Rounded Rectangles .....	117
Setting Line End Shape and Size .....	118
Setting Auto Scaling and Word Wrapping for a Text Box .....	119
Using Images .....	119
Placing an Image on the Canvas .....	119
Setting the Image Display Mode .....	119
Setting the Image Alignment .....	120
Setting the Image Color Transparency .....	120
Editing the Image .....	121
Setting the Image Editing Application .....	121
Selecting a Different Image .....	121
Using Buttons .....	122
Automatically Scaling Text in Buttons .....	122
Wrapping Text in Buttons .....	122
Configuring Buttons with Images .....	122
Editing Control Points .....	123
Moving Control Points .....	123
Adding and Removing Control Points .....	123
Changing the Tension of Curves and Closed Curves .....	124
Changing Angles of Arcs, Pies and Chords .....	124
Utilizing Sweep Angle Run-Time Properties .....	125
Monitoring and Showing Quality and Status .....	125
Using Status Elements .....	125
Setting Number Formats by Regional Locales .....	126
Design Time Considerations for Numeric Formatting .....	126
Enter Input Numbers in U.S. Format .....	127
Set the Regional Locale of the Computer Hosting the HMI/SCADA Software .....	129
Run-Time Considerations for Formatting Numbers .....	129
Restrictions of Numeric Formatting by Regional Locale .....	131
Numeric Strings Enclosed Within Quotation Marks .....	131

---

Numbers Passed as Script Parameters .....	132
Double-byte Character Languages .....	132
Using Windows Common Controls .....	132
Changing Background Color and Text Color of Windows Common Controls .....	133
Reading and Writing the Selected Value at Run Time .....	133
Configuring Radio Button Group Controls .....	134
Setting the 3D appearance of a Radio Button Group Control .....	134
Setting the Layout of the Radio Button Group Options .....	134
Using Radio Button Group-Specific Properties at Run Time .....	135
Configuring Check Box Controls .....	135
Setting the Default State of a Check Box Control .....	135
Setting the Caption Text of a Check Box Control .....	135
Setting the 3D appearance of a Check Box Control .....	135
Configuring Edit Box Controls .....	136
Setting the Default Text in an Edit Box Control .....	136
Configuring the Text to Wrap in an Edit Box Control .....	136
Configuring the Text to be Read-Only in an Edit Box Control .....	136
Configuring Combo Box Controls .....	137
Setting the Type of Combo Box Control .....	137
Setting the Width of the Drop-Down List .....	137
Avoiding Clipping of Items in the Simple Combo Box Control .....	138
Setting the Maximum Number of Items to Appear in the Combo Box Drop-Down List .....	138
Using Combo Box-Specific Properties at Run Time .....	138
Configuring Calendar Controls .....	138
Setting the Number of Calendar Month Sheets .....	139
Setting the First Day of the Week .....	139
Showing or Hiding Today's Date on a Calendar Control .....	139
Setting Title Fill Color and Text Color on a Calendar Control .....	140
Setting the Text Color for Trailing Dates in a Calendar Control .....	140
Setting the Default Value of the Calendar Control .....	141
Configuring DateTime Picker Controls .....	141
Configuring List Box Controls .....	142
Avoiding Clipping of Items in the List Box Control List .....	143
Using a Horizontal Scroll Bar in a List Box Control .....	143
Using List Box-Specific Properties at Run Time .....	143
Using Custom Properties .....	145
About Custom Properties .....	145
Managing Custom Properties .....	145
Adding and Deleting Custom Properties .....	146
Configuring Custom Properties .....	146
Validating Custom Properties .....	147
Clearing the Configuration of Custom Properties .....	147
Renaming Custom Properties .....	148
Linking Custom Properties to External Sources .....	148
Overriding Custom Properties .....	148
Reverting to Original Custom Property Values .....	148
Animating Graphic Elements .....	151
About Animations .....	151
Adding an Animation to an Element .....	151
Reviewing which Animations are Assigned to an Element .....	151



Showing and Hiding the Animation List .....	152
Removing Animations from an Element .....	152
Enabling and Disabling Animations .....	152
Validating the Configuration of an Animation .....	153
Clearing the Configuration from an Animation .....	153
Managing Animations .....	153
Organizing the Animation List .....	154
Switching between Animations .....	154
Configuring Common Types of Animations .....	154
Configuring a Visibility Animation .....	155
Configuring a Fill Style Animation .....	155
Configuring a Boolean Fill Style Animation .....	155
Configuring a Truth Table Fill Style Animation .....	156
Configuring a Line Style Animation .....	157
Configuring a Boolean Line Style Animation .....	157
Configuring a Truth Table Line Style Animation .....	158
Configuring a Text Style Animation .....	159
Configuring a Boolean Text Style Animation .....	160
Configuring a Truth Table Text Style Animation .....	160
Configuring a Blink Animation .....	161
Configuring an Alarm Border Animation .....	162
Understanding Requirements of Alarm Border Animations .....	162
Understanding the Behavior of Alarm Border Animations .....	163
Configuring Alarm Border Animation .....	165
Configuring Optional Alarm Border Animation Characteristics .....	166
Configuring a Percent Fill Horizontal Animation .....	168
Configuring a Percent Fill Vertical Animation .....	170
Configuring a Horizontal Location Animation .....	171
Configuring a Vertical Location Animation .....	172
Configuring a Width Animation .....	172
Configuring a Height Animation .....	173
Configuring a Point Animation .....	173
Configuring an Orientation Animation .....	174
Configuring a Value Display Animation .....	175
Configuring a Boolean Value Display Animation .....	175
Configuring an Analog Value Display Animation .....	176
Configuring a String Value Display Animation .....	177
Configuring a Time Value Display Animation .....	177
Configuring a Name Display Animation .....	179
Configuring a Tooltip Animation .....	179
Configuring a Disable Animation .....	180
Configuring a User Input Animation .....	180
Configuring a User Input Animation for a Discrete Value .....	180
Configuring a User Input Animation for an Analog Value .....	181
Configuring a User Input Animation for a String Value .....	182
Configuring a User Input Animation for a Time Value .....	183
Configuring a User Input Animation for an Elapsed Time Value .....	184
Configuring a Horizontal Slider Animation .....	185
Configuring a Vertical Slider Animation .....	185
Configuring a Pushbutton Animation .....	186
Configuring a Pushbutton Animation for a Boolean Value .....	186

Configuring a PushButton Animation for an Analog Value .....	187
Configuring a PushButton Animation for a String Value .....	188
Configuring an Action Script Animation .....	189
Configuring an Action Script Animation with a "Mouse-Down" Event Trigger .....	190
Configuring a Show Symbol Animation .....	190
Configuring a Hide Symbol Animation.....	197
Configuring a Hyperlink Animation.....	197
Configuring Element-Specific Animations .....	198
Configuring Animation for a Status Element .....	198
Restrictions of the Status Element .....	198
Configuring a Radio Button Group Animation .....	199
Configuring a Static Radio Button Group Animation .....	199
Configuring an Array Radio Button Group Animation.....	199
Configuring an Enum Radio Button Group Animation .....	200
Configuring a Check Box Animation .....	201
Configuring an Edit Box Animation .....	201
Configuring a Combo Box Animation .....	202
Configuring a Static Combo Box Animation .....	202
Configuring an Array Combo Box Animation.....	203
Configuring an Enum Combo Box Animation .....	203
Configuring a Calendar Control Animation .....	204
Configuring a DateTime Picker Animation .....	204
Configuring a List Box Animation.....	206
Configuring a Static List Box Animation.....	206
Configuring an Array List Box Animation .....	207
Configuring an Enum List Box Animation .....	207
Configuring a Trend Pen .....	208
Understanding the Types of Trend Plots .....	208
Understanding the Types of Trend Pen Periods.....	208
Submitting the Value Changes .....	209
Format Strings in Element-Specific Animations .....	209
Numbers .....	209
Dates .....	210
Enumerations .....	212
Format String Examples.....	212
Cutting, Copying and Pasting Animations .....	212
Substituting References in Elements .....	213
<b>Adding and Maintaining Graphic Scripts .....</b>	<b>215</b>
About Graphic Scripts .....	215
Predefined and Named Scripts .....	215
Execution Order of Graphic Scripts.....	215
Graphic Script Time outs (already in ITAA Integration Guide) .....	216
Security in Graphic Scripts.....	216
Error Handling.....	216
Signature Security for Acknowledging Alarms .....	216
SignedAlarmAck() Run-time Behavior.....	217
SignedAlarmAck() Scripting Tips .....	218
SignedAlarmAck() Applied Example.....	218
Configuring the Predefined Scripts of a Graphic .....	219
Ensuring Proper OnShow Script Execution .....	220

Adding Named Scripts to a Graphic .....	221
Editing Graphic Scripts .....	221
Renaming Scripts in a Graphic .....	222
Removing Scripts from a Graphic .....	222
Substituting Attribute References in Scripts .....	222
Example of Changing Element Properties using Scripts .....	223
Using Methods in Scripting .....	223
Configuring Edit Box Methods .....	223
Configuring Combo Box and List Box Methods .....	224
Adding and Inserting Items into a List.....	224
Deleting Items from a List.....	225
Finding an Item in a List .....	225
Reading the Caption of a Selected Item in a List .....	225
Associating Items with Values in a List .....	226
Loading and Saving Item Lists.....	226
<b>Using Client Controls .....</b>	<b>229</b>
About Client Controls .....	229
Organizing Client Controls .....	229
Embedding Client Controls .....	230
Viewing and Changing the Properties of Client Controls .....	230
Binding Client Control Properties to Attributes or Element References .....	231
Configuring Client Control Event Scripts .....	232
Animating Client Controls .....	232
Including Dynamically Loaded Assemblies with the Client Control .....	233
Requirements for Both Inclusion Methods .....	233
Sample XML for a Dynamically Loaded Assembly List.....	233
XML Schema for the Dynamically Loaded Assembly List .....	234
Embedding the XML Manifest Resource in the Primary Assembly .....	234
Including the XML Manifest Resource in an External Configuration File .....	234
Preventing Dynamically Loaded Assembly Import Issues .....	235
<b>Embedding Graphics within Graphics.....</b>	<b>237</b>
Embedding Graphics.....	237
Renaming Source Graphics and Hosting Objects .....	238
Editing the Embedded Graphic .....	239
Overriding Custom Properties of the Source Graphic.....	239
Restoring an Embedded Graphic to the Original Size of its Source Graphic .....	240
Converting an Embedded Graphic to a Group .....	240
Detecting the Source Graphic of an Embedded Graphic .....	240
Editing the Source of an Embedded Graphic.....	241
Controlling Size Propagation of Embedded Graphics.....	241
Setting the Anchor Point of a Source Graphic.....	241
Showing or Hiding the Anchor Points of Embedded Graphics.....	242

Enabling or Disabling Dynamic Size Change of Embedded Graphics .....	242
Selecting Alternate Graphics and Instances .....	243
Selecting Alternate Graphics .....	243
Selecting Alternate Instances .....	243
Detecting and Editing the Containing Object Instance .....	243
Creating a New Instance of the Containing Object .....	244
<b>Working with the Show/Hide Graphics Script Functions .....</b>	<b>245</b>
About the Show/Hide Graphic Functions .....	245
Configuring the Show/Hide Graphic Script Functions .....	245
Using the Display Graphic Browser and Display Automation Object Browser .....	246
Show/Hide Graphic Script Functions Guidelines .....	246
Using the Show/Hide Script Parameters and Properties .....	247
Using the Identity Property in the ShowGraphic() Function .....	247
Height and Width Aspect Ratio .....	247
Incompatible GraphicInfo Properties .....	248
Run Time Behavior of the Show/Hide Graphic Functions .....	252
Behavior of ShowGraphic Windows with the Same Identity .....	253
Closing a Graphic .....	253
Show/Hide Graphic Script Tips and Examples .....	253
Using Predefined and Named Scripts .....	254
Container Script Scenario .....	254
Working with Modal Windows .....	255
Using Hierarchical References and Containment Relationships .....	256
Scripting the Owning Object .....	257
Owning Object Scenario 1 .....	258
Owning Object Scenario 2 .....	260
Assigning Custom Property Values of a Graphic .....	261
Scripting Multiple Symbols .....	262
Multiple Symbols Scenario 1 .....	262
Multiple Symbols Scenario 2 .....	263
Multiple Graphics Scenario 3 .....	264
Multiple Graphics Scenario 4 .....	264
<b>Working with Symbol Wizards .....</b>	<b>265</b>
Introduction .....	265
Understanding the Symbol Wizard Editor .....	265
Understanding Choice Groups and Choices .....	266
Understanding Symbol Wizard Layers .....	266
Defining Graphic Configuration Rules .....	267
Examples of Graphic Configuration Rules .....	268
Designing a Symbol Wizard .....	268
Creating Graphic Choice Groups, Choices, and Options .....	269
Assigning Graphic Configuration Rules .....	270
Updating Graphic Layers .....	270
Associating Configuration Elements to Graphic Layers .....	271
Associating Graphic Elements to Graphic Layers .....	272
Using Shortcut Menu Commands to Edit Graphic Layer Graphic Elements .....	272
Associating Custom Properties to Graphic Layers .....	274
Associating Named Scripts to Graphic Layers .....	274

Verifying Graphic Configurations .....	275
Using Symbol Wizards in an Application .....	276
Embedding Symbol Wizards .....	276
Symbol Wizard Tips and Examples .....	277
Creating Visual Configurations of an Industrial Graphic .....	277
Planning Symbol Wizard Configurations .....	278
Identify Graphic Elements .....	280
Build a Visual Representation of a Symbol Wizard .....	281
Assign Graphic Elements, Named Scripts, and Custom Properties to Graphic Layers ....	283
Specify Rules to select Graphic Layers .....	283
<b>List of Element Properties .....</b>	<b>287</b>
Alphabetical List of Properties .....	287
List by Functional Area .....	308
Graphic Category Properties .....	309
Appearance Category Properties .....	309
Fill Style Group Properties .....	320
Line Style Group Properties .....	323
Text Style Group Properties .....	324
Runtime Behavior Group Properties .....	325
Custom Properties Group Properties .....	328
Order of Precedence for Property Styles .....	329
<b>Windows Common Control List Methods .....</b>	<b>331</b>
Overview of Windows Common Control List Methods .....	331
<b>QuickScript References .....</b>	<b>335</b>
Script Functions .....	335
Graphic Client Functions .....	335
GetCPQuality() .....	335
GetCPTimeStamp() .....	335
HideGraphic() .....	336
HideSelf() .....	336
Logoff() .....	337
ShowGraphic() .....	337
ShowLoginDialog() .....	347
Math Functions .....	347
Abs() .....	348
ArcCos() .....	349
ArcSin() .....	349
ArcTan() .....	349
Cos() .....	350
Exp() .....	350
Int() .....	350
Log() .....	351
Log10() .....	351
LogN() .....	352
Pi() .....	352
Round() .....	352
Sgn() .....	353
Sin() .....	353
Sqrt() .....	354
Tan() .....	354

Trunc()	354
Miscellaneous Functions	355
ActivateApp()	355
DateTimeGMT()	356
IsBad()	356
IsGood()	357
IsInitializing()	357
IsUncertain()	358
IsUsable()	358
LogCustom()	359
LogDataChangeEvent()	359
LogError()	360
LogMessage()	360
LogTrace()	361
LogWarning()	361
SendKeys()	362
SetAttributeVT()	364
SetBad()	364
SetGood()	365
SetInitializing()	365
SetUncertain()	365
SignedAlarmAck()	366
SignedWrite()	370
WriteStatus()	373
WWControl()	374
String Functions	374
DText()	374
StringASCII()	375
StringChar()	375
StringCompare()	376
StringCompareNoCase()	376
StringFromGMTTimeToLocal()	377
StringFromIntg()	378
StringFromReal()	378
StringFromTime()	379
StringFromTimeLocal()	380
StringInString()	381
StringLeft()	381
StringLen()	382
StringLower()	383
StringMid()	383
StringReplace()	384
StringRight()	385
StringSpace()	385
StringTest()	386
StringToIntg()	387
StringToReal()	387
StringTrim()	388
StringUpper()	389
Text()	389
WWStringFromTime()	390
System Functions	390
CreateObject()	391
Now()	391
QuickScript .NET Variables	391
Numbers and Strings	393

QuickScript .NET Control Structures.....	394
IF ... THEN ... ELSEIF ... ELSE ... ENDIF .....	394
IF ... THEN ... ELSEIF ... ELSE ... ENDIF and Attribute Quality .....	395
FOR ... TO ... STEP ... NEXT Loop.....	396
FOR EACH ... IN ... NEXT .....	397
TRY ... CATCH .....	397
WHILE Loop .....	398
QuickScript .NET Operators.....	399
Parentheses ( ) .....	400
Negation ( - ).....	401
Complement ( ~ ) .....	401
Power ( ** ) .....	401
Multiplication ( * ), Division ( / ), Addition ( + ),Subtraction ( - ) .....	401
Modulo (MOD) .....	401
Shift Left (SHL), Shift Right (SHR) .....	401
Bitwise AND ( & ) .....	402
Exclusive OR (^) and Inclusive OR (   ) .....	402
Assignment ( = ).....	402
Comparisons ( <, >, <=, >=, ==, <> ).....	402
AND, OR, and NOT.....	402





# CHAPTER 1

## About AVEVA Industrial Graphics

Industrial Graphics are graphics you can create to visualize data in an HMI/SCADA system.

You use the Industrial Graphic Editor to create Industrial Graphics from basic elements, such as rectangles, lines, and text elements. You can also use the Industrial Graphic Editor to embed and configure an Industrial Graphic from the Graphic Toolbox library of graphics.

After you create an Industrial Graphic, you can embed it into another graphic or an HMI system window and use it at run time.

You can embed an Industrial Graphic in a template or instance of an object providing several ways to visualize object-specific information quickly and easily. Embedding a graphic in a template means that you can update one graphic and cascade the changes throughout your application.

Depending on your development requirements, you can select where and how to store industrial graphics.

- Create and store graphics as a standard set that you can re-use, such as a generic valve graphic.
- Store graphics as templates if you want to use the graphics in multiple instances at run time.
- Store graphics for use in a specific application.

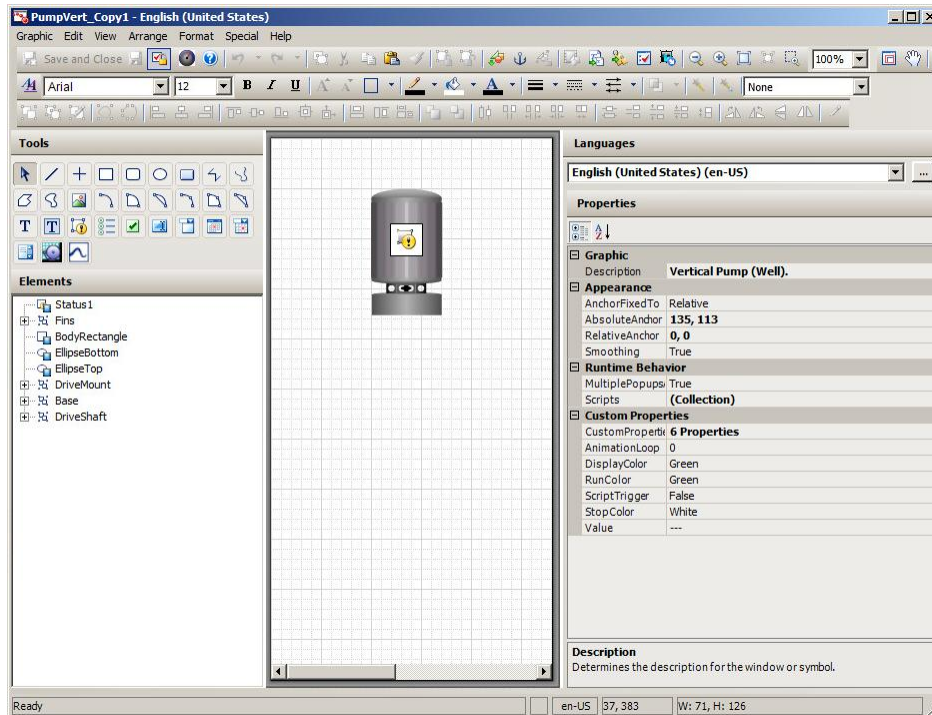
## The Industrial Graphic Editor

The Industrial Graphic Editor allows you to create an Industrial Graphic. It also allows you to open an existing graphic created in your HMI/SCADA software for editing.

To create an Industrial Graphic in the editor, select a basic graphical object, called an element, from the Tools panel and place it on the drawing area, called the canvas. Typical elements are lines, rectangles, ellipses, curves, and so on.

Then, change the appearance of drawn elements by accessing their properties directly or by graphically manipulating them. You can also change the appearance of an embedded Industrial Graphic by configuring the associated custom properties. Finally, you can configure animations for the elements or the graphics.

After you open the Industrial Graphic Editor, you will see the various tools and palettes to create and customize graphics.



The Industrial Graphic Editor includes the following areas:

- **Tools Panel:** A collection of elements you use to create your graphic.
- **Canvas:** The area in which you place and edit elements to create a graphic.
- **Elements List:** List that displays named elements on the canvas in a hierarchical view.
- **Language Selector:** List that displays the configured languages for the graphic. For more information, see *Switching Languages for Graphic Elements*.
- **Properties Editor:** Shows the properties belonging to one or more currently selected elements.
- **Animation Summary:** Area that shows you a list of animations belonging to the currently selected element. It is only visible if an element is selected.
- **Symbol Wizard Editor:** The Symbol Wizard Editor is a feature of the Industrial Graphic Editor to create graphics containing multiple visual and functional configurations called Symbol Wizards. For more information, see *Creating Multiple Configurations of a Graphic* on page 46.

## Tools Panel

The Tools panel contains elements you can select to create your graphic on the canvas.



The Tools panel includes:

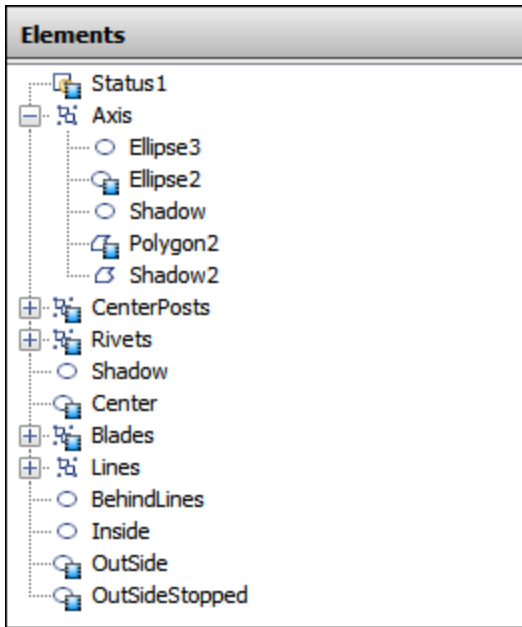
- Basic objects, such as lines, rectangles, polygons, arcs, and so on.
- A pointer tool to select and move elements on the canvas.
- Windows controls, such as combo boxes, calendar controls, radio button groups, and so on.
- A status element that you can use to show quality and status of a selected attribute.

For more conceptual information, see [Elements](#).

For more information on how to use elements, see *Working with Graphic Elements* on page 57.

## Elements List

The Elements List is a list of all elements on the canvas.



The Elements List is particularly useful for selecting one or more elements that are visually hidden by other elements on the canvas. Use the Elements List to:

- See a list of all elements, groups of elements, and embedded graphics on the canvas.
- Select elements or groups of elements to work with them.
- Rename an element or a group of elements.

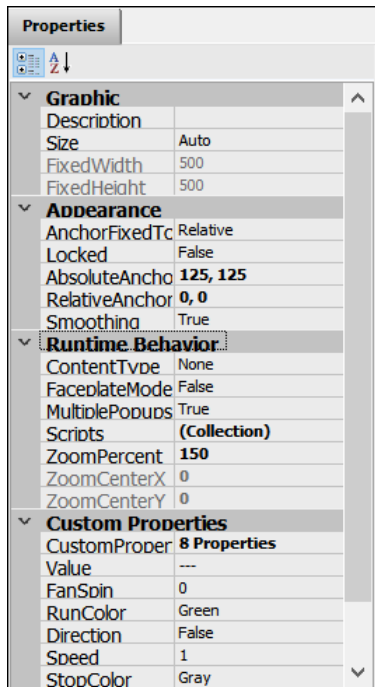
---

**Caution:** If you rename an element or a group, the animation references to it do not automatically update. You must manually change all animation links referencing the old name. For more information, see *Substituting References in Elements* on page 213.

---

## Properties Editor

Use the Properties Editor to view and set properties for the selected element or group of elements.

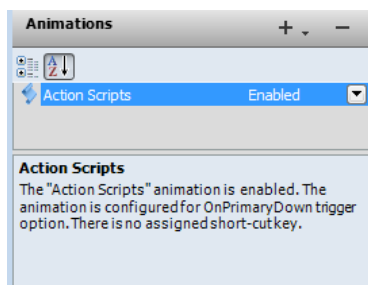


For more conceptual information about element properties, see *Properties* on page 25.

For more information on how to use element properties, see *Editing Common Properties of Elements and Graphics* on page 95.

## Animations Summary

Use the Animations summary to review, select, and configure the animation behavior of a selected element.



For an overview of the different animation types, see *Animation Types* on page 27.

For more information on how to use the animations, see *Animating Graphic Elements* on page 151.

## Canvas

The canvas is your drawing area. Use it like any image editing software by drawing elements and changing them to suit your requirements.

## Elements

You use elements to create a graphic. The Industrial Graphic Editor provides the following:

- Basic elements such as lines, rectangles, ellipses, arcs, and so on
- Status element to show a quality status icon
- Windows controls, such as combo boxes, calendar controls, radio button groups, and so on

You can create the following from existing elements on the canvas:

- Groups
- Path graphics

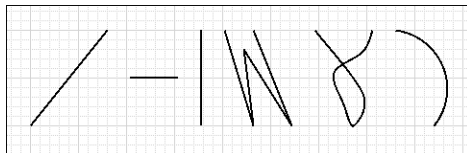
You can embed the following on the canvas:

- Imported Client Controls
- Other graphics

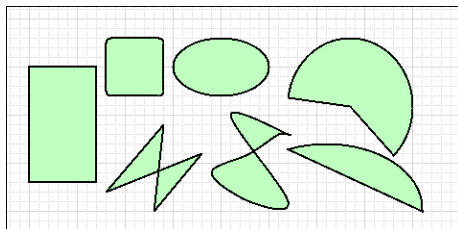
## Basic Elements

You can use the following basic elements to create a graphic:

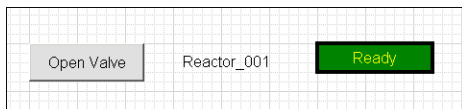
- ◆ Open elements, such as lines, H/V lines, polylines, curves, and arcs.



- Closed elements, such as rectangle, rounded rectangle, ellipse, polygon, closed curve, pie, and chord. You can draw arcs, pies, and chords from two points or from three points.



- Text elements, such as buttons, text, and text boxes.

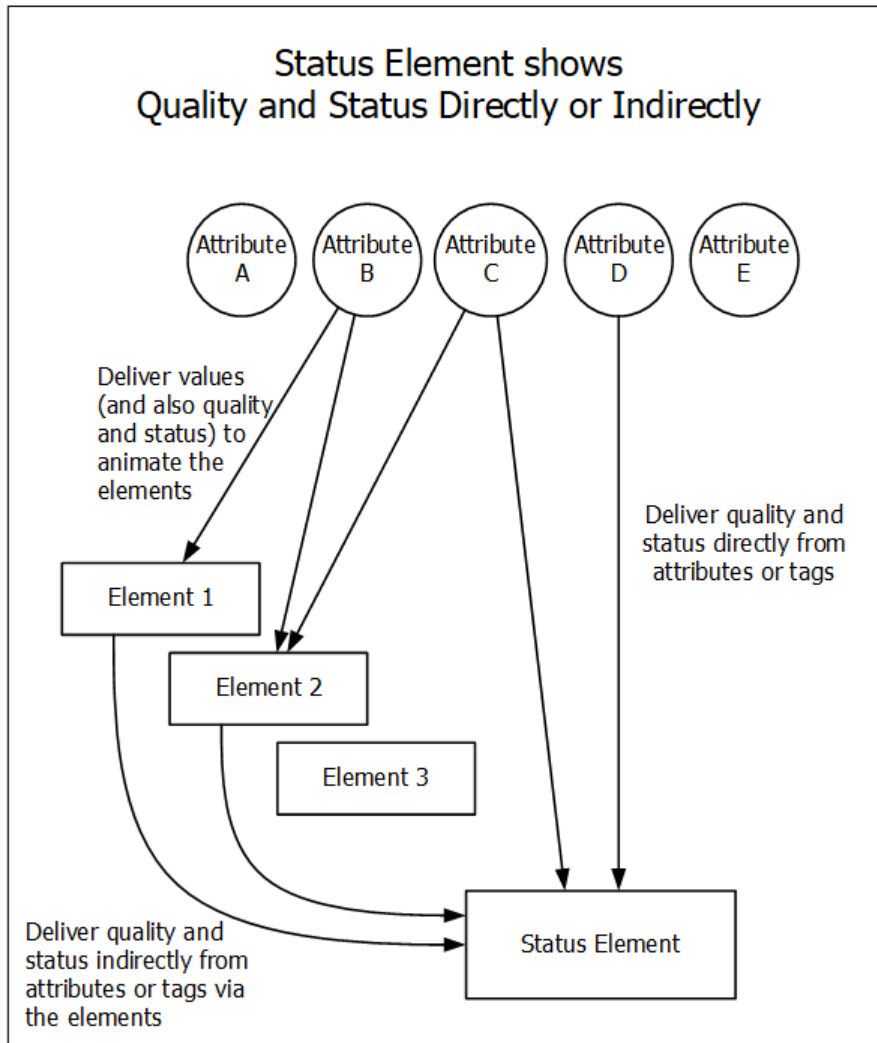


## Status Element

The status element provides a graphical representation of the communications status of an attribute or a tag, and the data quality of the attribute's or tag's value. Use the status element to monitor and indicate communications status and data quality of:

- All attributes or tags used in one or more specified animated elements at the same hierarchical level.

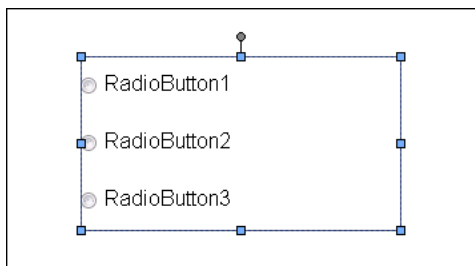
- One or more specified attributes or tags.



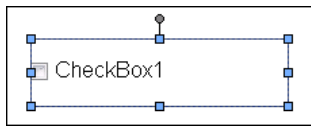
## Windows Common Controls

Using Windows common controls, you can add extended user interaction to your graphic. You can use:

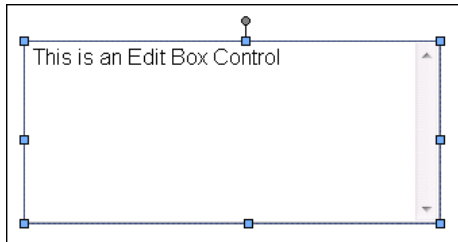
- A radio button group to select an option from a mutually exclusive group of options.



- A check box to add a selectable option.



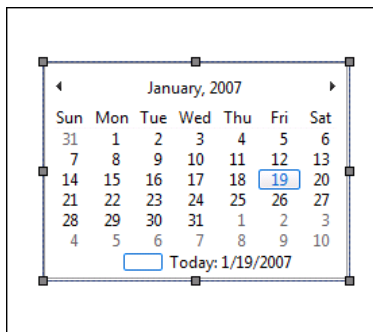
- An edit box to add an entry box for text.



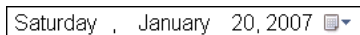
- A combo box to select an option from a drop-down list.



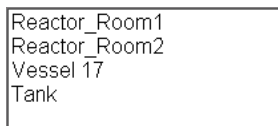
- A calendar to use a date selection control.



- A date and time picker to select a date and time in a compact format.



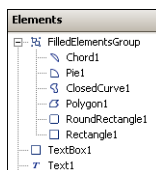
- A list box to select one or more options from a list.



## Groups

Grouping enables you to combine elements as a unit. Groups can contain elements and other groups.

Groups are shown in the Elements List with a default name, such as Group1. They are shown as a branch in the element hierarchy.



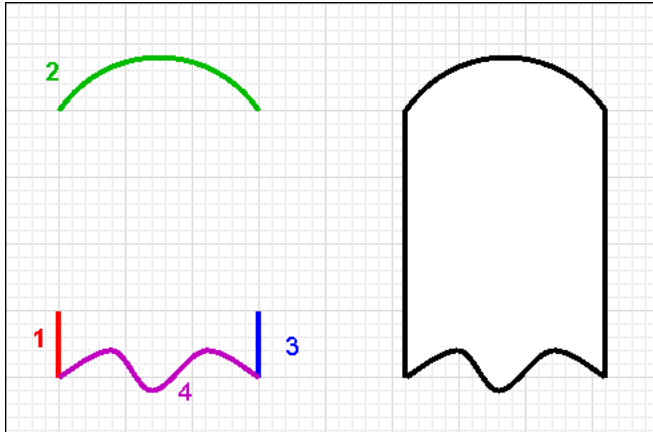
For example, you can create a series of elements that model a valve in your facility. When the valve has all the properties and animations you want, you can group the elements together.

You can then work with the elements as one set of elements or, by selecting the elements in the Elements List, you can work with the individual elements in the group without having to break the group. This is called inline editing.

Another advantage of inline editing is that you can easily select an individual element graphically without having to know its element name.

## Path Graphics

Path graphics are elements that combine selected open elements, such as lines, H/V lines, polylines, curves, and arcs, into a single closed graphic element.



A path graphic depends on the:

- Order in which you drew the elements. Each element is linked to the next element by z-order. The z-order of the elements is the order shown in the Elements List.
- Direction in which you drew its elements. The ending point of one element is connected to the starting point of the next element.

The properties of the elements contained within a path graphic are retained. When you break a path graphic, the elements it contains appear as they did before you created the path graphic.

A path graphic has the same properties as a rectangle, ellipse, or polygon. These properties are lost when you break the path.

## Windows Client Controls

Windows client controls are .NET-based controls you can use in an Industrial graphic to extend its functionality.

After you embed a client control into a graphic, you can:

- Connect the native properties of the client control to attributes or tags, and element references.
- Configure scripts for client control properties.
- Edit the native properties directly with the Properties Editor.
- Configure and override animations.

If your HMI supports it, you can embed a graphic that contains an embedded client control into your application and use the functionality of the client control directly in your HMI.

For more information, see *Using Client Controls* on page 229.



## Properties

Properties determine the appearance and behavior of an element or the graphic. For example, the width property determines the width in pixels of the selected element.

There are two types of properties:

- Predefined properties
- Custom properties

### Predefined Properties

Properties are specific to the selected element and can vary between elements of different types. All elements have the following property categories:

- Graphic - the name of the element (or group)
- Appearance - element dimension, location, rotation, transparency, and locked status

You can view specific properties for a specific kind of element or group by clicking a drawing tool and drawing an element.

You set properties at design time. Some properties can be read or written to at run time, such as X, Y, Width, Height, Visible, and so on. The element type determines which properties are available and can be read or written at run time.

### Custom Properties

Use custom properties to extend the functionality of a graphic. A custom property can contain:

- A value that can be read and written to.
- An expression that can be read.
- An object attribute that can be read and written to if the attributes allows being written to.
- A property of an element or graphic.
- A custom property of a graphic.
- A reference to a tag.

For example, for a tank graphic called TankSym you can create a custom property called TankLevel that is calculated from an attribute reference to Tank\_001.PV. You can then reference the tank level by TankSym.TankLevel.

Custom properties appear in the Properties Editor when no elements are selected. You can edit default initial values of custom properties in the editor directly or use the **Edit Custom Properties** dialog box to do so.

For more information, see *Using Custom Properties* on page 145.

### Properties of Groups

Groups have their own properties you can view and set in the Properties Editor. For most properties, changing group properties indirectly affects the properties of its contained elements.

You can change the following group properties:

- Name (Name)
- Position (X, Y)
- Size (Width, Height)

- Orientation (Angle)
- Point of Origin (AbsoluteOrigin, RelativeOrigin)
- Transparency (Transparency)
- Locked (Locked)
- Enablement (Enabled)
- Tab Order (TabOrder)
- Tab Stop (TabStop)
- Single Object Treatment (TreatAsIcon)
- Visibility (Visible)

### Changing/Renaming a Group Name

If you change the group name, it has no affect on the contained elements. The contained elements keep their name.

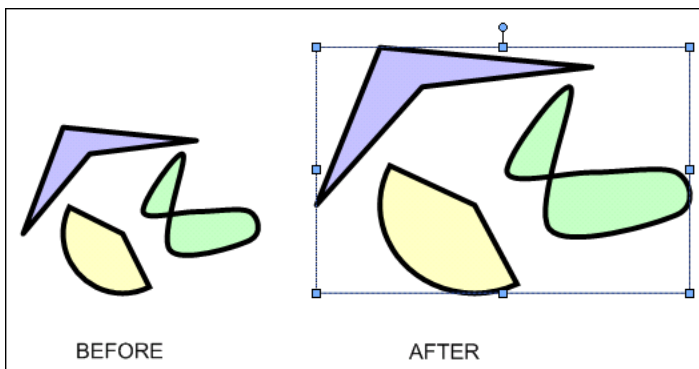
If you rename an element or a group, the animation references to it are not automatically updated. You must manually change all animation links referencing the old name. For more information, see *Substituting References in Elements* on page 213.

### Changing the Position of a Group

If you change the position of the group, all contained objects are moved with the group. They maintain the relative position to each other, but their absolute positions change.

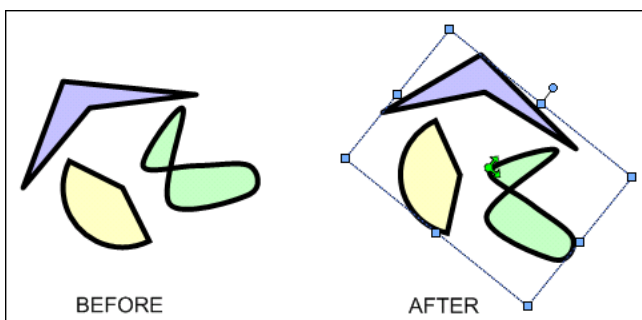
### Changing the Size of a Group

If you change the size of the group, all contained objects are resized proportionally.



### Changing the Orientation of a Group

If you change the angle of the group, all contained objects are rotated with the group around the origin of the group, so that the group remains visually intact.



## Changing the Transparency of a Group

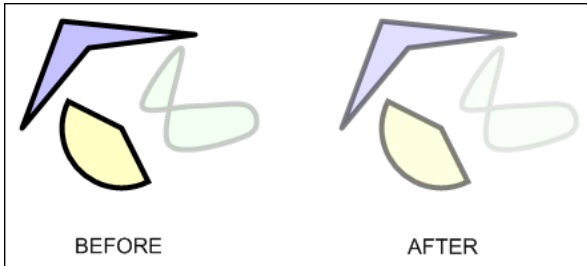
If you increase the transparency of the group, all contained objects appear more transparent, but their own transparency property values do not change. If you change their transparency values, it is in relation to the transparency level of the group.

For example, if you add an element with 80 percent transparency to a group, and then apply 50 percent transparency to the group, the element appears to have 90 percent transparency.

This is calculated as follows:

$$1 - (1 - 0.8) * (1 - 0.5) = 0.9$$

The transparency property values, however, stay unchanged at 80 percent for the element and 50 percent for the group.



## Locking the Group

If you lock the group, it has no effect on the contained elements. You can still edit the contained elements in inline editing mode. You cannot move, resize, or rotate the group.

## Run-Time Properties of a Group

If you change the run-time properties of a group, the elements do not inherit the property value of the group, but they do inherit the behavior of the group.

For example, if you create a group from elements, some of which have their visibility set to true and some to false, then set the group visibility to false, ALL elements in that group are invisible.

However the Visible property values of the contained elements still maintain their original values (true or false).

## Renaming a Group or its Elements

If you rename an element or a group, the animation references to it are not automatically updated. You must manually change all animation links referencing the old name. For more information, see *Substituting References in Elements* on page 213.

## Animations

You can use animations to bind the run-time behavior and appearance of elements to attributes and tags, custom properties, and other element's properties.

For example, you can bind the vertical fill of a rectangle to an tag or attribute that contains the current level of a tank.

Animations are specific to the selected element and vary between elements of different types.

## Animation Types

There are two types of animations:

- Visualization animations determine the element's appearance, such as blinking, fill style, percent fill horizontal, value display, and so on.

- Interaction animations determine the element's behavior, such as horizontal sliders, user input, and so on.

There are visualization and interaction animations that are specific to certain elements. For example, the `DataStatus` animation is specific to the `Status` element. Element-specific animations also determine element behavior and appearance.

You can configure the following common animation types:

<b>Animation Type</b>	<b>Description</b>
<b>Visibility</b>	Shows or hides the element depending on a value or an expression.
<b>Fill Style</b>	Specifies the interior fill style depending on a discrete or analog expression or one or more conditions.
<b>Line Style</b>	Specifies the style and pattern of the element line depending on a discrete or analog expression or one or more conditions.
<b>Text Style</b>	Specifies the style of the element text depending on a discrete or analog expression or one or more conditions.
<b>Blink</b>	Sets the element to blink invisibly or with specified colors depending on a discrete value or expression.
<b>Element Style</b>	Defines a set of visual properties that determine the appearance of text, lines, graphic outlines, and interior fill shown in Industrial graphics.
<b>% Fill Horizontal</b>	Fills the element with color partially from left to right or vice versa, depending on an analog value or expression.
<b>% Fill Vertical</b>	Fills the element with color partially from top to bottom or vice versa, depending on an analog value or expression.
<b>Location Horizontal</b>	Positions the element with a horizontal offset depending on an analog value or expression.
<b>Location Vertical</b>	Positions the element with a vertical offset depending on an analog value or expression.
<b>Width</b>	Increases or decreases the element width depending on an analog value or expression.
<b>Height</b>	Increases or decreases the element height depending on an analog value or expression.
<b>Point</b>	Changes the X and Y coordinate values of one or more selected points on a graphic or graphic element.
<b>Orientation</b>	Rotates the element by an angle around its center point or any other point depending on an analog value or expression.
<b>Value Display</b>	Shows a discrete, analog, string value, time value, name or expression.
<b>Tooltip</b>	Shows a value or expression as a tooltip when the mouse is moved over the element.

<b>Animation Type</b>	<b>Description</b>
<b>Disable</b>	Disables the element's animation depending on a Boolean value or expression.
<b>User Input</b>	Enables the run-time user to type a Boolean, analog, string, time or elapsed time value that is then assigned to an attribute.
<b>Slider Horizontal</b>	Enables the run-time user to drag the element left or right and write back the offset to an analog attribute.
<b>Slider Vertical</b>	Enables the run-time user to drag the element up or down and write back the offset to an analog attribute.
<b>Pushbutton</b>	Writes predetermined values to Boolean or analog references when the user clicks on the element.
<b>Action Scripts</b>	Runs an action script when the run-time user clicks on the element.
<b>Show Symbol</b>	Shows a specified graphic at a specified position when the run-time user clicks on the element.
<b>Hide Symbol</b>	Hides a specified graphic when the run-time user clicks on the element.

## Data Sources for Animations

The data used for animations can come from various sources. You can configure the animation to point at these sources. Animation data can come from:

- Attributes of AutomationObjects.
- Predefined properties of an element or graphic.
- Custom properties of a graphic.
- HMI tags.

## Animation Capabilities of Groups

By default, a group of elements has limited animation capabilities of its own. For a group you can configure the following animations:

- Blinking
- Enabling/disabling
- Vertical and horizontal location
- Orientation
- Height and width
- Visibility

However, you can set the TreatAsIcon property value to True. The group is then treated as a single object and you can configure more animations. These animations take precedence over animations defined for the elements within the group.

## Animation States

Some animations have multiple configuration panels.

A state selection panel appears, where you can select the animation state. Depending on what you select, the configuration panel is populated differently. The animation state can be a:

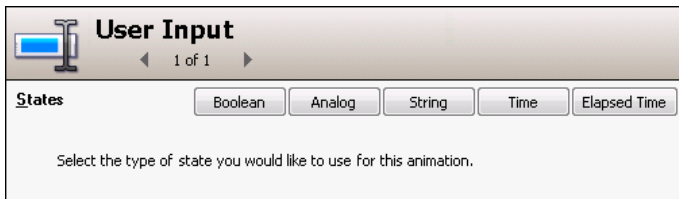
- Data type, where the animation is tied to a specific data type.
- Truth table, where the animation is tied to a set of Boolean conditions.

### Data Type Animation State

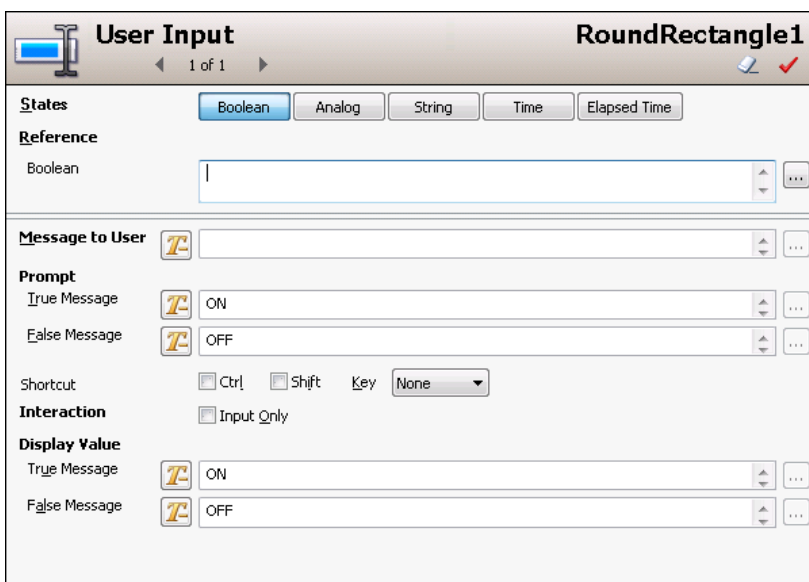
Certain animations support configuration of one or more data types. In the configuration panel of an animation, select the data type you want to configure, such as:

- Boolean
- Analog
- String
- Time
- Elapsed Time
- Name

For example, if you select the **User Input** animation link, the **User Input** state selection page appears on the right in the **Edit Animations** dialog box.



A configuration panel appears below the **States** buttons. For example, a configuration panel that is specific to the user input of a Boolean value.



## Truth Table Animation State

Certain animations support the configuration of a truth table. The truth table is a collection of up to 100 Boolean conditions you can configure to determine the output.

You can configure the default appearance for the case that none of the conditions are fulfilled.

The conditions are evaluated from top to bottom of the list. When the first true condition is met, its assigned appearance is the one used and the condition evaluation stops.

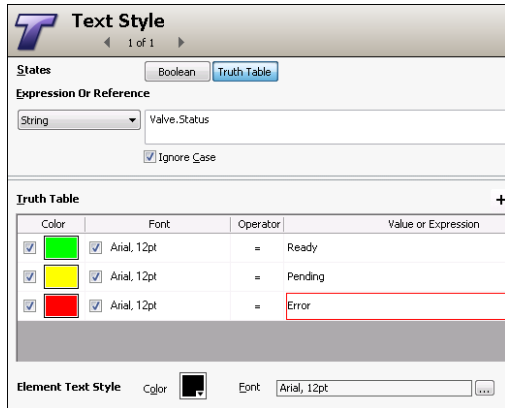
For example, you want a text animation to use a different text color depending on the value of a string attribute, such as a status indicator.

Status indicator	Text color
Ready	Green
Pending	Yellow
Error	Red

If you select the **Text Style** animation link, the **Text Style** state selection page appears on the **Edit Animations** dialog box.



You can click the **Truth Table** button to configure conditions for the appearance of the text style.



By default the text color is black if none of the conditions are fulfilled at run time.

## Embedding Graphics

You can embed graphics into other graphics. Embedding graphics enables you to rapidly develop more complex graphics with common components.

For example, you can create a single tank graphic, then embed it multiple times in another graphic to create a graphic representing a collection of tanks.

There is no limit to the number of levels of embedding.

Embedded graphics appear in the Elements List. The default name is the same as the source graphic, followed by a numeric sequence.

## Changing Embedded Graphics

After you embed a graphic, you can change its size, orientation or transparency. You can add a limited set of animations to the graphic, such as:

- Visibility
- Blink
- Horizontal and vertical location
- Width and height
- Orientation
- Disable
- Touch Pushbuttons (Discrete Value, Action, Show Window, and Hide Window)

You can configure its public custom properties, if any exist.

You cannot:

- Change the graphic definition of the embedded graphic from within the hosting graphic.
- Embed a graphic contained in an object created in your HMI/SCADA software into a graphic that is contained in the Industrial Graphic Editor.
- Create circular references. A circular reference occurs when one graphic (Graphic A) has embedded within it another graphic (Graphic B) that has embedded within it a graphic that directly or indirect references back to the first graphic (Graphic A).

You can, however, change the embedded graphic by changing its source graphic. The changes you make propagate to the embedded graphic.

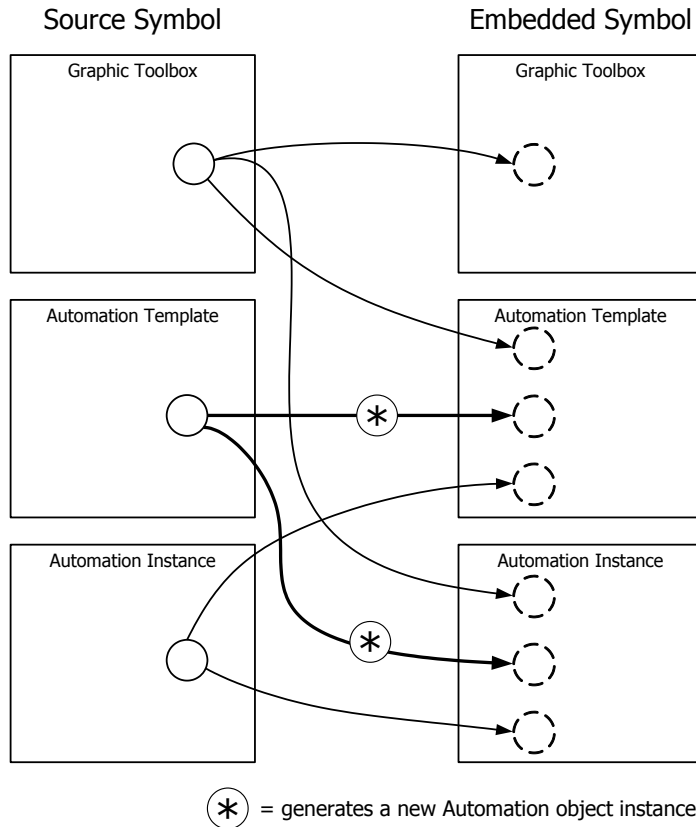
## Embedding and Instantiation

You can embed graphics into graphics, object templates, or an object instance if supported by your HMI/SCADA software. You can store a graphic as a template for use in multiple instances, or create a single instance of the graphic from the template for specific use.

You can embed graphics contained in a template into graphics contained in other objects. When you do so, a new object instance is created. You can give it a name, and the new instance inherits the graphic, but does not contain it.

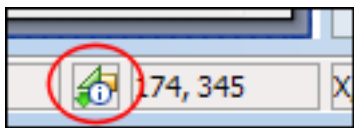


You can only embed graphics contained in an object instance into graphics contained in other objects. The template or instance inherits the graphic, but does not contain it.



## Graphic Change Propagation

When you make changes and save a graphic, the changes are propagated to any other graphic that embed that graphic. The Industrial Graphic Editor shows an icon in the status bar below the canvas that indicates that the source of an embedded graphic changed.



You can accept the change immediately or when you open the graphic again.

---

**Important:** You cannot undo/redo the modifications done for a graphic whose source has changed and the propagated change is accepted.

---

When a graphic is changed, its external size can also be changed. Industrial Graphics support dynamic size propagation and anchor points that determine how and if size changes are propagated. For more information about size propagation, see *Size Propagation and Anchor Points* on page 34.

If the graphic is edited:

- All graphics hosted by templates and instances that contain embedded instances of this graphic are also updated.
- All embedded instances of this graphic are also marked for an update.

If the graphic is hosted by an AutomationObject and edited:

- All graphics hosted by derived AutomationObjects are also updated.
- All embedded instances of this graphic in an HMI application hosted by derived AutomationObjects are marked for an update.

## Size Propagation and Anchor Points

An anchor point controls how changes in graphic size are propagated to embedded instances. By default, the anchor point of the graphic is the center point of all elements on the canvas.

This can be done graphically on the canvas, or by setting anchor position properties in the Properties Editor.

There are two types of anchors:

- Use the AbsoluteAnchor property to specify its position as absolute coordinates.
- Use the RelativeAnchor property to specify its position as coordinates relative to the graphic center.

When you embed a graphic, the embedded graphic inherits the anchor point in relation to its own center point.

You can also set the AnchorFixedTo property. When you make changes to the graphic that affects its size, the AnchorFixedTo property determines if the absolute position or relative position of the anchor point is recalculated. This property can have following values:

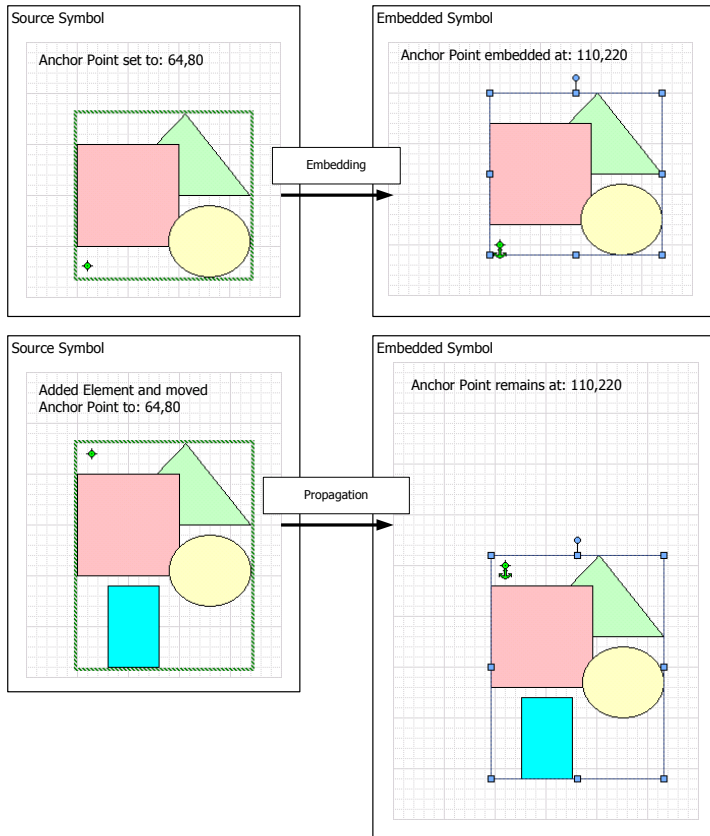
- Absolute: The absolute anchor point position is unchanged, and the relative anchor point position is recalculated.
- Relative: The absolute anchor point position is recalculated, and the relative anchor point position is unchanged.

---

**Note:** When you change the AbsoluteAnchor property, the AnchorFixedTo property is set to the value Absolute. When you change the RelativeAnchor property, the AnchorFixedTo property is set to the value Relative.

---

You can change the position of the anchor point of the graphic. This affects the position of the embedded instances. The anchor points of the embedded instances, however, remain unchanged.



**Note:** You can change the anchor point of an embedded graphic. This moves the embedded graphic. It does not change the anchor point position in relation to the graphic. You can resize or rotate the embedded graphic. The anchor point moves in relation to the embedded graphic. You can also use the AnchorPoint property in the Properties Editor to change the position.

## Estimating Graphic Performance

You can assess the performance of an Industrial graphic at runtime using the Graphics Performance Index (GPI). The GPI calculates the estimated call up time when the graphic you are building in the Industrial Graphic Editor is launched at run time.

Call up time pertains to the interval between the time the user or system makes a request to show the pertinent graphic and the graphic appearing on the screen with live data. The calculation is based on contents of the graphic launched at run time.

## Estimating Graphics' Performance

Use the Graphics Performance Index (GPI) window to view estimated performance statistics of a graphic you are building or editing.

**Note:** The Graphics Performance Index window can also be viewed if using the Symbol Wizard in Preview mode, and for graphics currently in a non-editable state.

### To estimate graphic performance using the Graphics Performance Index

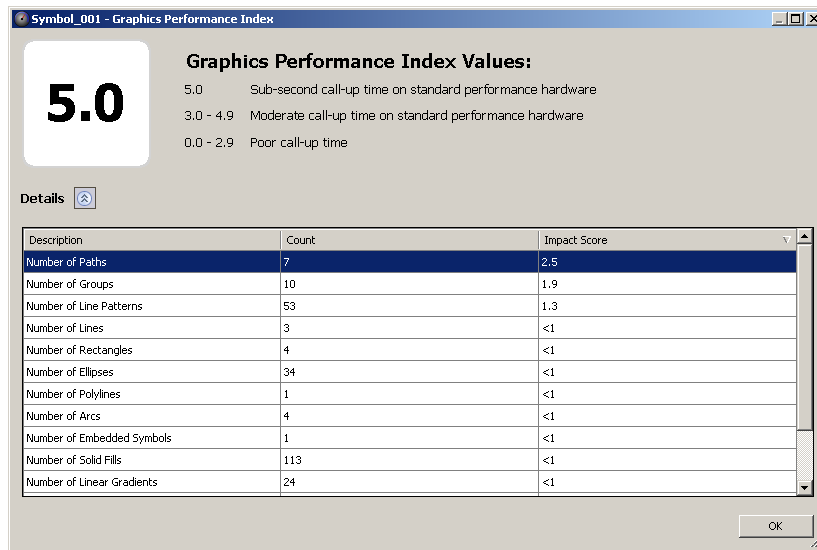
1. Do any of the following in the Industrial Graphic Editor:

- On the **Graphic** menu, click **Graphics Performance Index**.



- Click the Graphics Performance Index icon on the toolbar.
- Click the **GPI:** label in the status bar.
- Press Ctrl+P on your keyboard.
- Press P on your keyboard if the Graphics menu is open.

The **Graphics Performance Index** window appears.



The GPI rating appears in the upper left corner of the Graphics Performance Index window. This calculation is a figure in the range from 0 to 5, based on the type and number of components included in the graphic. A rating of 5 indicates a graphic call up time less than 1 second. See *Understanding GPI Rating Calculations* on page 37 for details about how the GPI rating is calculated.

---

**Note:** The GPI calculation is based on results from an ideal environment in which required subscriptions are made to an engine running on scan and appropriate references are established.

---

- Click the **Details** button to expand the window. A list displays showing rows of details for component types in the graphic that are greater than 0 in size. These details are as follows:

Column Header	Relevance
Description	Component type
Count	Number of items comprising the component type
Impact Score	Call up time in milliseconds for the component type

Rows are sorted in descending order by Impact Score. You can re-sort rows by clicking the designated column header.

- After reviewing the contents, click **OK**. You can edit the graphic and test the GPI again using this tool.

## Understanding GPI Rating Calculations

GPI rating calculations are based on components in a graphic.

All visible graphics on the screen are counted, except for Symbol Wizard graphics, that may be stored in memory, which reduces the amount of content loaded and rendered at run time. Exceptions to Symbol Wizard graphic component calculations are documented for the pertinent categories in this section.

A table for each of the following component categories contains pertinent counter types and corresponding quantities and measured times in milliseconds: Elements, Animations, Styles, Reference, Custom Property, and Scripts.

### Elements Category

Graphic element components are counted individually. Though Symbol Wizard graphics are not counted, if any graphic element in a Symbol Wizard graphic is set to be visible at design time, it will be counted at run time.

The following table shows a list of element component types and the score assigned to each item, based on the estimated amount of time for processing the specified quantity of each component type:

Element Counter Type	Counter Description	Number of Items	Impact Score
Line	Number of Lines	50000	4098
Rectangle	Number of Rectangles	50000	1113.4
RoundRectangle	Number of Rounded Rectangles	50000	1652.8
Ellipse	Number of Ellipses	50000	1381
Button	Number of Buttons	50000	3142.2
PolyLine	Number of Polylines	50000	6278.4
Curve	Number of Curves	50000	7980.2
Polygon	Number of Polygons	50000	4465.2
ClosedCurve	Number of Closed Curves	50000	7414.6
Image	Number of Images	5000	14568.4
Arc	Number of Arcs (2 and 3 points)	50000	6500.2
Pie	Number of Pies (2 and 3 points)	50000	4696.2
Chord	Number of Chords (2 and 3 points)	50000	3798.8
Text	Number of Texts	50000	11575.6
TextBox	Number of Text Boxes	50000	5526.2

<b>Element Counter Type</b>	<b>Counter Description</b>	<b>Number of Items</b>	<b>Impact Score</b>
Status	Number of Statuses	25000	4013.6
RadioButton	Number of Radio Buttons	500	3487.6
CheckBox	Number of Check Boxes	500	7955.4
EditBox	Number of Edit Boxes	500	1557.2
ComboBox	Number of Combo Boxes	500	4744.4
Calendar	Number of Calendars	500	11729.8
DateTime	Number of Date Times	500	3566.8
ListBox	Number of List Boxes	500	4166
EmbeddedSymbol	Number of Embedded Symbols	50000	9760.8
Group	Number of Groups	50000	9631.2
Path	Number of Paths	50000	17765.8
TrendPen	Number of Trend Pens	2000	3847.4

## Animations Category

Animation components are counted individually. The following table shows a list of animation component types and the score assigned to each item, based on the estimated amount of time for processing the specified quantity of each component type:

<b>Animation Counter Type</b>	<b>Counter Description</b>	<b>Number of Items</b>	<b>Impact Score</b>
UserInput	Number of User Input Animations	50000	5231.8
LineStyle	Number of Line Style Animations	50000	1980.4
FillStyle	Number of Fill Style Animations	50000	2363
TextStyle	Number of Text Style Animations	50000	5034.2
PercentFill	Number of Percent Fill Animations (Vertical and Horizontal together)	50000	5610.8

<b>Animation Counter Type</b>	<b>Counter Description</b>	<b>Number of Items</b>	<b>Impact Score</b>
ValueDisplay	Number of Value Display Animations	50000	3054.4
Orientation	Number of Orientation Animations	50000	3776
Visibility	Number of Visibility Animations	50000	1290.6
Disable	Number of Disable Animations	50000	1256.8
ShowGraphic	Number of ShowSymbol Animations	50000	4240.6
HideGraphic	Number of HideSymbol Animations	50000	5001
Location	Number of Location Animations (Vertical and Horizontal together)	50000	3204.4
Size	Number of Size Animations (Vertical and Horizontal together)	50000	2907
ActionScript	Number of Action Script Animations	50000	9329
Slider	Number of Slider Animations (Vertical and Horizontal together)	50000	3091.6
Tooltip	Number of Tooltip Animations	50000	2480
PushButton	Number of Push Button Animations	50000	1076.4
Blink	Number of Blink Animations	50000	11349
PointAnimation	Number of Point Animations	50000	10220
NamedStyle	Number of Element Style Animations	50000	6726.8
AlarmAnimation	Number of Alarm Border Animations	50000	14796.8

## Styles Category

Style components are counted individually. The following table shows a list of style component types and the score assigned to each item, based on the estimated amount of time for processing the specified quantity of each component type:

Style Counter Type	Counter Description	Number of Items	Impact Score
SolidFill	Number of Solid Fills (Fill or Line usage)	50000	50
PatternFill	Number of Pattern Fills (Fill or Line usage)	50000	127.8
TextureFill	Number of Texture Fills (Fill or Line usage)	50000	10330.8
LinearGradient	Number of Linear Gradients	50000	547.8
RadialGradient	Number of Radial Gradients	50000	1337.8
Transparencies	Number of Transparencies	50000	30
LinePattern	Number of Line Patterns	50000	1203.6
LineEnd	Number of Line Ends	50000	2117.8

### Reference Category

Reference components are counted individually. The following table shows a list of reference component types and the score assigned to each item, based on the estimated amount of time for processing the specified quantity of each component type:

Reference Counter Type	Counter Description	Number of Items	Impact Score
ExternalReference	Number of External References	2000	1942.2
CustomPropReference	Number of Custom Property References	50000	3658.2
RuntimePropReference	Number of Runtime Property References	50000	7417

### Custom Properties Category

Custom property components are counted individually. Though Symbol Wizard graphics are not counted, if any named custom property in a Symbol Wizard graphic is set to be visible at design time, it will be counted at run time.

The following table shows a list of custom property component types and the score assigned to each item, based on the estimated amount of time for processing the specified quantity of each component type:

Custom Properties Counter Type	Counter Description	Number of Items	Impact Score
CustomProperty	Number of Custom Properties	50000	3020



Custom Properties Counter Type	Counter Description	Number of Items	Impact Score
CustomPropertyOverridden	Number of overridden Custom Properties	50000	3403

## Scripts Category

OnShow and Action scripts are counted individually. Container scripts, which include While Showing, OnHide, and named scripts, are counted together. Though Symbol Wizard graphics are not counted, if any named script in a Symbol Wizard graphic is set to be visible at design time, it will be counted at run time.

The following table shows a list of script component types and the score assigned to each item, based on the estimated amount of time for processing the specified quantity of each component type:

Scripts Counter Type	Counter Description	Number of Items	Impact Score
OnShowSmallScript	Number of scripts with 10 lines or less	50000	5989.8
OnShowMediumScript	Number of scripts with over 10 lines and under 50 lines	50000	17026
OnShowLargeScript	Number of scripts with 50 lines and over	50000	54274
ActionScripts	Number of Action Scripts	50000	9329
ContainerScripts	Number of Container Scripts	25000	7978.8

## Examining a Graphic with a 4.5 GPI Rating

The following table shows values for components in a sample graphic that received a GPI rating of 4.5:

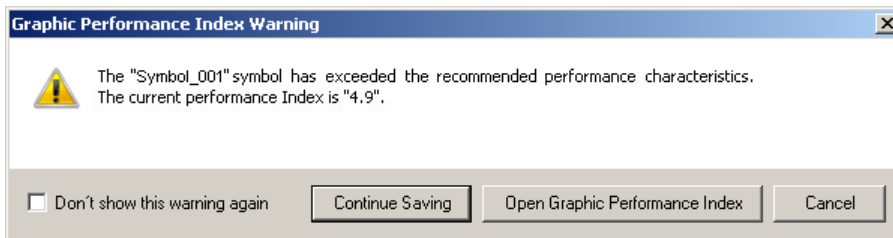
Category	Performance Counter	Config. Count	Processing Capacity per second	Projected Time (sec.)
Elements	Number of Lines (basic style, solid colors, no transparency)	20	3000	0.01
	Number of Curves	30	1000	0.03
	Number of Text elements with strikethrough or underline style (significant impact due to the expansive draw text API)	10	500	0.02
	Number of Paths	40	60	0.67

<b>Category</b>	<b>Performance Counter</b>	<b>Config. Count</b>	<b>Processing Capacity per second</b>	<b>Projected Time (sec.)</b>
	Number of Groups	20	400	0.05
	Number of Embedded Graphics	5	350	0.01
	Max level of nesting	3	500	0.01
	Number of elements with transparency	20	500	0.04
	Number of Calendar elements	0	20	0.00
Animations	Number of Percent Fill Animations	2	1000	0.00
	Number of animations with a truth table	5	1000	0.01
	Number of other animations	10	1000	0.01
Styles	Number of Linear Gradients	20	3000	0.01
	Number of Radial Gradients	30	1000	0.03
	Number of Line Ends	20	400	0.05
	Number of non-solid line styles	5	350	0.01
Reference	Number of External References	2	1000	0.00
	Number of local Custom Property References	5	1000	0.01
	Number of external Custom Property References	10	1000	0.01
	Number of element Custom Property References	0	1000	0.00
CustomProperty	Number of Custom Properties	20	1000	0.02
	Number of overridden Custom Properties	5	1000	0.01
Scripts	Number of scripts with less than 10 statements	2	1000	0.00
	Number of scripts with 11 - 50 statements	5	1000	0.01
	Number of scripts with more than 50 statements	10	1000	0.01
	Number of reference expressions	5	1000	0.01

Category	Performance Counter	Config. Count	Processing Capacity per second	Projected Time (sec.)
			<b>Total</b>	<b>1.70</b>
			<b>GPI</b>	<b>4.50</b>

## Saving a Graphic that May Impact Runtime Performance

When building a graphic or attempting to save a graphic with a GPI rating less than 5.0, the **Graphic Performance Index Warning** window appears with information about the GPI rating for the graphic.



You can perform the following tasks in the **Graphic Performance Index Warning** window:

- Click **Continue Saving** to save the graphic without additional edits.
- Click **Open Graphic Performance Index** to open the **Graphics Performance Index** window.
- Click **Cancel** to close the **Graphic Performance Index Warning** window.
- Select the **Don't show this warning again** check box to prevent this window from displaying for this graphic in the future.

**Note:** The option to hide or show this warning window can also be configured in the Graphic Symbol Designer Preferences window. For more information, see *Configuring Designer Preferences* on page 52.

## Showing Quality and Status

To show a specified status or quality at run time, you can:

- Use a *Status element* (see "*Status Element*" on page 21) that shows you an icon. It indicates the status or quality of specified attributes or tags directly or those used indirectly in elements.
- Change the appearance of animated elements based on the status and quality of attributes or tags they use.

## Showing Quality and Status with the Status Element

The Status element cannot monitor attributes of:

- Elements that are not in the same hierarchy level in the Elements List.
- Elements that use the attributes in scripts.
- Elements that are invisible at run time.

For more information on how to configure status on an element, see *Configuring Animation for a Status Element* on page 198.

For more information on how to configure the appearance of a status element, see [Setting the Appearance of a Status Element](#).

## Showing Quality and Status by Overriding

You can override the appearance of animations depending on its configured attributes by:

- Overriding the animation or changing the appearance of the element.
- Drawing an outline around the element.

Overriding the appearance of animations also applies to:

- Elements contained in groups.
- Elements in graphics embedded in other graphics.

Overriding the appearance of animations does not apply to:

- Elements that use the monitored attribute in scripts.
- Elements that are invisible at run time.

For more information, see [Overriding Element Appearance Depending on Quality and Status of its Attributes](#).

# CHAPTER 2

## Managing Industrial Graphics

### Creating a New Graphic

You can create a new graphic:

- In your HMI graphics repository, such as the Graphic Toolbox, for generic graphics that you frequently use in different situations. For example, a valve graphic.
- From a template, if object and application templates are supported by your HMI/SCADA software. Do this if you want to re-use the graphic in combination with the object functionality. An example is a graphic representing a specific tank and your production facility has multiple tanks.
- From an object if instantiation is supported by your HMI/SCADA software. Do this if you are unlikely to re-use the graphic in any other situation.

Creating a new graphic using your HMI/SCADA software will add it to your graphics library. You can then manage and use your graphics.

- Rename
- Copy
- Move
- Edit the graphic
- Create toolsets - groups of graphics for use in a particular visualization application or to represent a group of related devices
- Move graphics between toolsets
- Rename toolsets

### Opening Graphics for Editing

### Organizing Graphics

Use the library available to your HMI/SCADA software to organize your graphics by creating a folder hierarchy as you would with files and folders in Windows Explorer. You can move graphics around within the folder hierarchy. These folders are called Graphic Toolsets.

### Importing and Exporting Graphics

You can import and export graphics to the graphic libraries associated with your HMI/SCADA software. Consult the corresponding user guide for specific information and procedures.

### Deleting a Graphic

You can delete a graphic that you no longer need. Deleting a graphic removes it completely from the Industrial Graphic Editor.

- When you delete a graphic, you are shown where the graphic is used. This allows you to understand the impact of deleting the graphic before you actually delete it.
- You cannot delete a graphic that someone else has open for editing or left checked out.

- If you delete a graphic that is embedded in another graphic, it shows a Not Found message.

**To delete a graphic**

1. Select the graphic you want to delete and click **Delete**. The **Delete** dialog box appears.
2. Review the places the graphic is being used, and then click **Yes**.

## Creating Multiple Configurations of a Graphic

The Symbol Wizard Editor is a feature of the Industrial Graphic Editor to create multiple configurations of a graphic. A graphic configuration represents different visual or functional variations of a graphic.

Graphic configurations are created using layers containing associated graphic elements, custom properties, and named scripts. Based on graphic properties and possible values of these properties, rules are applied that specify when a layer is part of a graphic configuration.

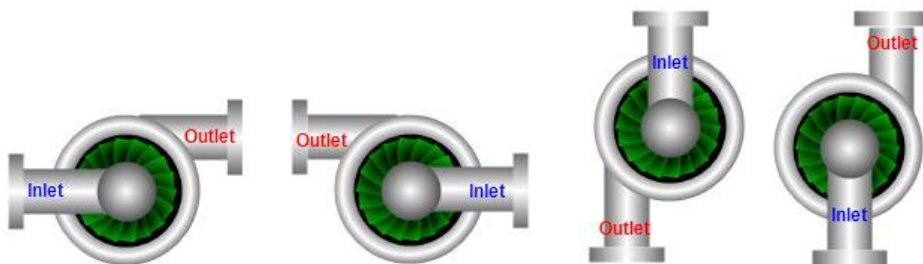
## Understanding Visual and Functional Graphic Configurations

Standard Industrial Graphics show reasonably realistic views of process objects. These graphics can be modified with the Symbol Wizard to incorporate multiple visual configurations in a graphic.

Situational Awareness Library graphics are a special set of Industrial Graphics that are available for use in your HMI/SCADA software, and are designed using the Symbol Wizard Editor. However, they are protected graphics and their design cannot be changed. However, you can select Wizard Options from the Symbol Wizard Editor to select the configurations that are incorporated into each graphic’s design.

### Visual Graphic Configurations

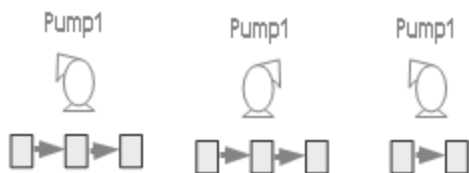
Using an example of a centrifugal pump with separate inlet and outlet pipes, there are four practical visual configurations. The pump’s blade housing is common and appears in all possible configurations. But, the pump’s inlet and outlet pipes can be placed at the left or right in a horizontal direction or at the top or bottom when the pump is oriented vertically.



Orientation is the visual property that identifies the different configurations of a pump graphic. The attributes associated with the Orientation property are left, right, top, and bottom.

### Functional Graphic Configurations

Situational Awareness Library symbols include functional properties in addition to visual properties. For example, a multi-stage pump symbol includes a Wizard Option to select either a five-stage, three-stage, or single stage pump in addition to a visual Orientation property to select left or right pump configurations.



## Embedding Graphics

You can embed graphics into other graphics. Embedding graphics enables you to rapidly develop more complex graphics with common components.

For example, you can create a single tank graphic, then embed it multiple times in another graphic to create a graphic representing a collection of tanks.

There is no limit to the number of levels of embedding.

Embedded graphics appear in the Elements List. The default name is the same as the source graphic, followed by a numeric sequence.

## Appearance of Embedded Graphics

Embedded graphics appear in the Elements List. The default name is the same as the source graphic, followed by a numeric sequence.

## Changing Embedded Graphics

After you embed a graphic, you can change its size, orientation or transparency. You can add a limited set of animations to the graphic, such as:

- Visibility
- Blink
- Horizontal and vertical location
- Width and height
- Orientation
- Disable
- Touch Pushbuttons (Discrete Value, Action, Show Window, and Hide Window)

You can configure its public custom properties, if any exist.

You cannot:

- Change the graphic definition of the embedded graphic from within the hosting graphic.
- Embed a graphic contained in an object created in your HMI/SCADA software into a graphic that is contained in the Industrial Graphic Editor.
- Create circular references. A circular reference occurs when one graphic (Graphic A) has embedded within it another graphic (Graphic B) that has embedded within it a graphic that directly or indirect references back to the first graphic (Graphic A).

You can, however, change the embedded graphic by changing its source graphic. The changes you make propagate to the embedded graphic.

## Viewing a Graphic in Read-Only Mode

You can view a graphic in read-only mode if you don't want to edit it, or if it is checked out by somebody else.

If you open a graphic in read-only mode, you have access to all functions in the Industrial Graphic Editor that do not change the graphic.

### To view a graphic in read-only mode

1. Select the graphic that you want to view in read-only mode.
2. Click **Open Read-Only**. The selected graphic opens in the Industrial Graphic Editor.





## CHAPTER 3

# Using the Industrial Graphic Editor

## Showing, Hiding and Adjusting Panels

You can edit graphics using the Industrial Graphic Editor. Depending on where the graphic is contained, you can start the Industrial Graphic Editor from your HMI/SCADA software:

You can:

- Show and hide Industrial Graphic Editor panels to allocate more space on the canvas.
- Pan and zoom the canvas to make finer or more granular adjustments to elements.
- Place a grid on the canvas surface to align elements more precisely.

You can hide the Properties Editor and Animation Summary to allocate more space on the canvas.

### To show or hide the Properties Editor and Animation Summary panels

- Do either of the following:
  - Press ALT + ENTER.
  - On the **View** menu, click **Properties**.

You can also adjust the size of the Elements List and Properties Editor.

### To adjust the size of panels

1. Drag the dividing line between the panels to specify the new panel size.
2. Release the mouse button and the panels are resized.

## Panning and Zooming the Canvas

You can pan and zoom the canvas to make finer visual adjustments to the elements or to get a better overview of a large graphic.

Use the Pan and Zoom toolbar to pan and zoom.



### Panning

You can use the Pan functions of the Pan and Zoom toolbar to do the following:

- Use the **Pan and Zoom** window to select which part of the canvas appears on the screen.
- Grab the canvas with the Hand tool and move it (Pan).

You can also use the scroll wheel of the mouse to pan up and down in the current canvas display.

### Using the Pan and Zoom Window to Pan

Use the **Pan and Zoom** window to pan the canvas area.

### To use the Pan and Zoom window for panning

1. On the Pan and Zoom toolbar, click the **Pan and Zoom** window icon. The **Pan and Zoom** window appears.
2. In the **Pan and Zoom** window, move the mouse within the red rectangle. The pointer hand icon appears.
3. Click and hold the left mouse button down.
4. Drag the mouse. The red rectangle moves with the mouse.
5. Release the mouse button. The area shown in the canvas is changed accordingly.

## Using the Hand Tool to Pan

Use the Hand tool to pan the canvas area. This is equivalent to picking up the canvas and moving it so that the visible canvas area changes.

### To use the Hand tool to pan



1. On the **Pan and Zoom** toolbar, click the **Pan** icon.
2. Move the mouse over the canvas. The Hand tool pointer appears.
3. Click the canvas to grab the canvas and keep the mouse button down.
4. Move the mouse to change the area of canvas that is shown.
5. Release the mouse button.

## Using the Mouse Scroll Wheel to Pan

You can use the mouse scroll wheel to:

- Pan up or down.
- Pan 360 degrees.

### To use the mouse scroll wheel to pan up or down

1. Click the canvas so that no elements are selected.
2. Move the mouse scroll wheel:
  - Forward to pan up.
  - Backward to pan down.

### To use the mouse scroll wheel to pan in any direction

1. Click the canvas so that no elements are selected.
2. Click the mouse scroll wheel. The pointer appears in 360 degrees scroll mode.
3. Move the mouse. The visible area of the canvas is panned accordingly.
4. When you are done, click the canvas.

## Zooming

Use the Pan and Zoom toolbar to:

- Zoom in on a specified point to magnify the current elements.
- Zoom out from a specified point.

- Zoom to the default zoom factor (100 percent).
- Zoom so that the currently selected element is shown across the available canvas area or zoomed to the maximum value of 500 percent.
- Zoom in on an area of the canvas using a "rubber band" selection with your mouse.
- Specify or select a zoom factor.

You can also use the CTRL key and the scroll wheel of the mouse to zoom in and zoom out the current canvas view.

## Zooming In to a Specified Point

You can zoom in by 25 percent of the default scale to any specified point on the canvas.

### To zoom in to a specified point



1. Click the Zoom In icon in the toolbar.
2. Move the mouse over the canvas. The Zoom In pointer appears.
3. Click the canvas to where you want to zoom in. The canvas is zoomed in at the specified point.

## Zooming Out from a Specified Point

You can zoom out by 25 percent of the default scale from any specified point on the canvas.

### To zoom out to a specified point

1. Click the Zoom Out icon in the toolbar.
2. Move the mouse over the canvas. The Zoom Out pointer appears.
3. Click the canvas from where you want to zoom out. The canvas is zoomed out from the specified point.

## Zooming to the Default Zoom Value

You can reset the zoom to the default zoom value 100 percent.

### To reset the zoom to the default zoom value



- Click the Zoom to Normal icon in the toolbar. The canvas zoom is reset to its default.

## Zooming a Selected Element

You can zoom one or more selected elements so that they appear as large as possible in the allocated canvas area. This is useful when you want to make fine adjustments to one or more elements.

### To zoom a selected element

1. Select the elements you want to zoom.
2. Click the Zoom To Selection icon in the toolbar. The visible canvas is zoomed so that the selected elements appear as large as possible.

## Zooming a Specified Area

You can zoom a specified area by using the "rubber band" selection method.

### To zoom a specified area

1. Click the Rubber Band Zoom icon.
2. Move the mouse over the canvas. The Rubber Band pointer appears.
3. Move the mouse to the top left corner of the area you want to zoom.
4. Hold the left mouse button down and then drag the mouse to the bottom right corner of the area you want to zoom.
5. Release the mouse button. The area is zoomed to the entire canvas area.

## Selecting or Specifying a Zoom Value

You can select a defined zoom value or type a zoom value. Valid values are 25 percent to 500 percent.

### To select or specify a zoom value

- On the Zoom and Pan toolbar, do one of the following:
  - Click the zoom value list and select a zoom value.
  - Click the zoom value in the zoom value list, type a valid value, and then click **Enter**.

## Using the Pan and Zoom Window to Change the Zoom

Use the **Pan and Zoom** window to change the zoom of the canvas.

---

**Note:** You can also use the **Pan and Zoom** window to "scroll" to a different part of the canvas. This is called panning. For more information, see *Panning* on page 49.

---

### To use the Pan and Zoom window for zooming

1. On the Zoom and Pan toolbar, click the **Pan and Zoom** window icon. The **Pan and Zoom** window appears.
2. In the **Pan and Zoom** window, move the mouse over a corner or an edge of the red rectangle.
3. Click and hold the left mouse button down. The corresponding resize pointer appears.
4. Drag the mouse. The red rectangle changes size proportionally.
5. Release the mouse button. The zoom of the canvas is changed accordingly.

## Using the Mouse Scroll Wheel for Zooming

You can use the mouse scroll wheel to zoom the canvas area. The canvas is then zoomed on the midpoint of all selected elements or, if none are selected, on the midpoint of the canvas.

### To use the mouse scroll wheel for zooming

- Press and hold the CTRL key and move the scroll wheel:
  - Forward to zoom in by a factor of 25 percent of the default zoom value.
  - Backward to zoom out by a factor of 25 percent of the default zoom value.

## Configuring Designer Preferences

Use the **Designer Preferences** dialog box to set Industrial Graphic Editor preferences. Preferences can be configured for the following:

- Grid Settings - The grid helps you precisely place and move elements on the canvas.
- Canvas Settings - The settings for the appearance of the graphic on the canvas can also be configured.

- Graphics Performance Index Warning window visibility
- Image Editor selection

### To open the Designer Preferences dialog window

1. Open the Industrial Graphic Editor.
2. On the **View** menu, click **Preferences**. The **Designer Preferences** dialog box appears.

### To configure Grid Settings

1. Click the box next to the **Grid color** label. The **Select Grid Color** dialog box appears. For more information, see *Setting a Solid Color* on page 103.
2. In the **Grid size** box, type a value from 1 to 100 to specify the distance in pixels between each line in the grid.
3. In the **Major subdivisions** box, type a value from 1 to 10 to specify the number of major subdivisions of the grid. Major subdivisions are emphasized lines that visually create larger grid cell blocks.
4. Clear or select the **Grid visible** check box to hide or show the grid.
5. Clear or select the **Snap to grid** check box. With the snap-to-grid option set, when you move elements or groups on the canvas they are moved to the closest grid intersection. If this option is not set, you can move the elements freely to any location on the canvas.

### To configure Canvas Settings and graphic appearance

1. Click the box next to the **Background Color** label. The **Select Canvas Color** dialog box appears. For more information, see *Setting a Solid Color* on page 103.
2. Clear or select the **Symbol Smoothing** check box. If this option is not set, lines drawn on the canvas may show jagged edges. With this option set, lines drawn on the canvas show smooth edges.
3. Clear or select the **Show Anchor** check box. With this option selected, the graphic displays anchor icons, if anchors were created in the graphic.

### To configure Graphic Performance Index Settings

- Clear or select the **Show the Graphic Performance Index warning on save** check box. If this option is disabled, the **Graphic Performance Index Warning** window will not appear when saving a graphic with a GPI rating calculated to be less than 5.0. For more information about the Graphics Performance Index, see *Estimating Graphic Performance* on page 35."

### To select an Image Editor

- Choose the graphic editing tool from the **Image Editor** menu. If you select **Choose Custom Editor**, the **Select Image Editing Application** window appears so you can make a selection.

### To save your settings as default settings

1. Click **Save as Default**.
2. Click **Apply**.
3. Click **OK**.

## Using the Symbol Wizard Editor

The Symbol Wizard Editor in the Industrial Graphic Editor allows you to create graphics with multiple configurations called Symbol Wizards. They create Symbol Wizards with the Symbol Wizard Editor. You can embed Symbol Wizards and use the Symbol Wizard Editor to select the configuration needed for an application.

Start creating a multi-configuration graphic by opening a graphic element or graphic in the Industrial Graphic Editor. Show the Symbol Wizard Editor by clicking the Symbol Wizard icon from the Industrial Graphic Editor's menu bar, selecting it as an option of the **View** menu, or pressing the Alt+W key combination.

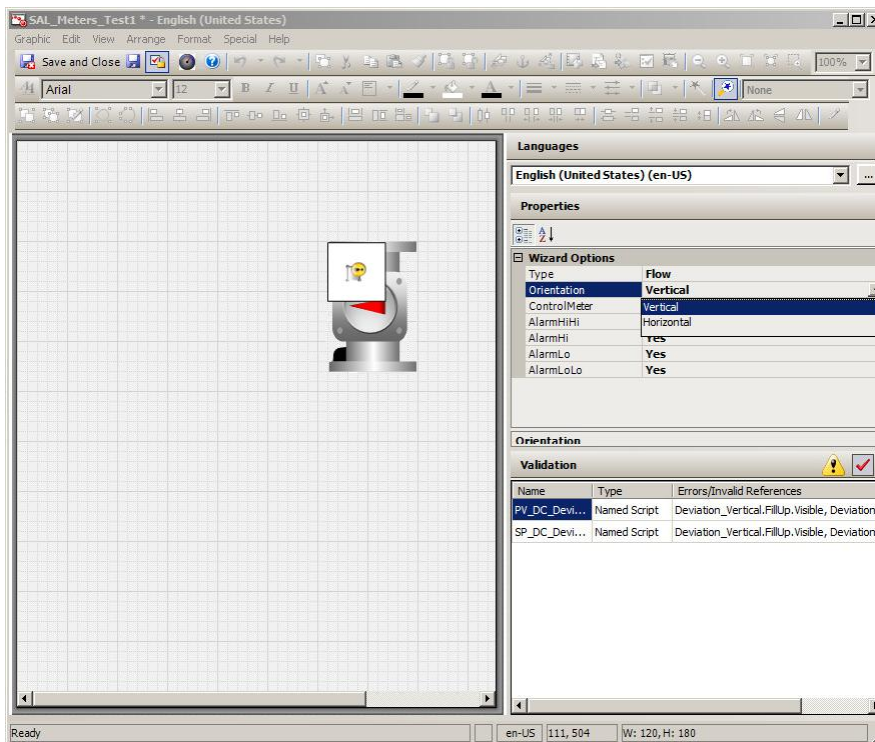
The Industrial Graphic Editor window updates to show tabbed Symbol Wizard panes at the left of the window. The top pane shows the graphic elements, named scripts, and custom properties of the graphic in separate views.

The bottom pane shows tabbed **Options** and **Layers** panes. The **Options** pane shows a hierarchical list of Choice Groups, Choices, and Options that define graphic properties and the possible values associated with each property. The **Layers** pane includes a list of defined graphic layers. Beneath each layer, separate folders contain the graphic's elements, custom properties, and named scripts associated with each layer. You can add, edit, or delete items associated with the **Options** and **Layers** panes.

Symbol Wizard **Option Properties** or **Layer Properties** panes appear to the right of the canvas area in the Industrial Graphic Editor window after selecting items from the **Options** or **Layers** panes.

Both properties pane shows the name of the selected item and any rule associated with the item. If a Choice Group is selected from the **Options** pane, the **Options Properties** pane also shows the default value of the Choice Group and a **Description** field.

After creating the configurations of a graphic with the Symbol Wizard Editor, use the Symbol Wizard Preview to verify that all configurations are correct. The Symbol Wizard Preview can be opened by clicking it from the menu bar, selecting it as an option of the **View** menu, or pressing the Alt+P key combination.

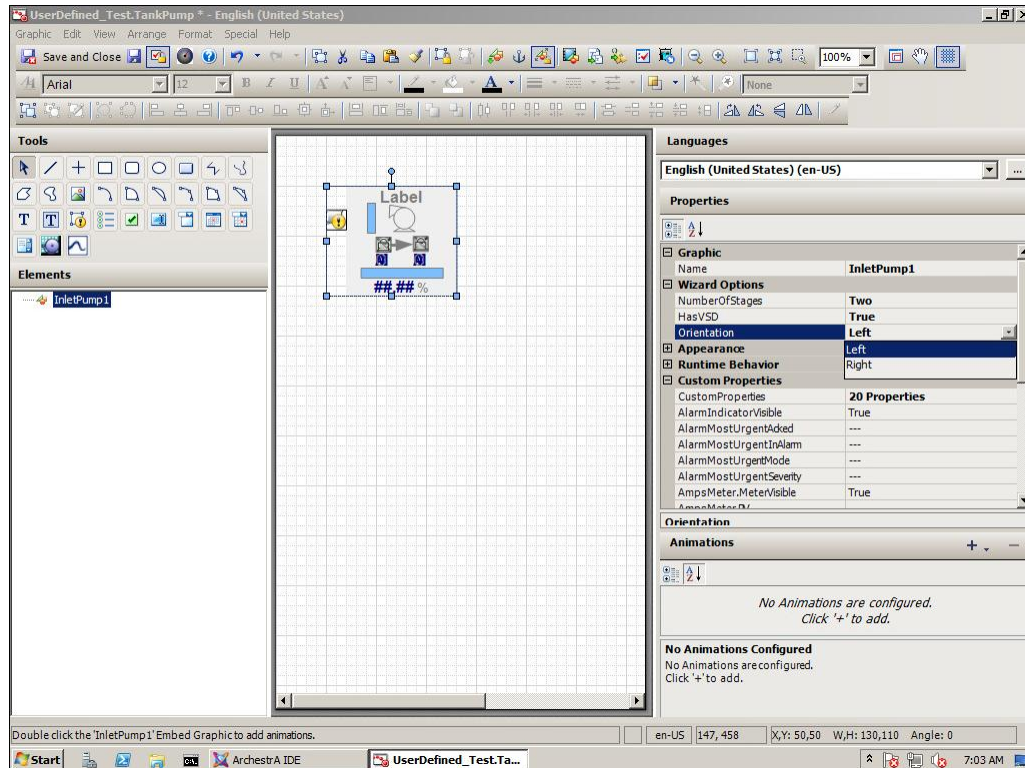


After opening Symbol Wizard Preview, the **Properties** pane shows **Wizard Options**, which includes drop-down menus to select options to show the different configurations created for the graphic. As options are selected, the graphic is updated to show the selected configuration.

The **Validation** pane shows any script or custom property errors within the graphic. Selecting a listed error from the **Validation** pane shows the **Custom Properties** or **Scripts** dialog box to identify and correct an error within the graphic.

After verifying that all graphic configurations are correct, save the graphic into the Graphic Toolbox. For more information about the Symbol Wizard tasks completed by a user, see *Designing a Symbol Wizard* on page 268.

To create an application containing Symbol Wizards, add a graphic to an automation object or create a new graphic from the Graphic Toolbox. Then, embed a Symbol Wizard. The graphic appears with the default configuration selected.



The **Wizard Options** pane shows a set of drop-down lists with configuration options. Select options from the drop-down lists to change the graphic's configuration to meet the needs of an application. Finally, edit and update the custom properties and named scripts that are associated with the multi-configuration graphic. For more information about the Symbol Wizard tasks, see *Using Symbol Wizards in an Application* on page 276.





# CHAPTER 4

## Working with Graphic Elements

### About Graphic Elements

This section explains how to work with the common features of graphic elements. For information about features specific to certain elements such as element properties, see *Editing Graphic-Specific and Element-Specific Properties* on page 117.

Graphic elements are basic shapes and controls you can use to create a graphic to your specifications. You can:

- Draw an element by selecting an element from the Tools panel, placing it on the canvas, and then configuring its properties.

Select one or more elements on the canvas with the mouse or from the **Element** list.

- Edit certain elements in a special way called inline editing.
- Copy, cut, paste, and duplicate elements.
- Move elements around on the canvas.
- Align elements to each other.
- Change the spacing between elements.
- Resize elements.
- Change the z-order of elements to change which elements appear on top of others when they overlap.
- Rotate elements.
- Change the origin of elements to specify around which point the elements are rotated.
- Flip elements on their horizontal or vertical axis.
- Lock elements to stop them being moved or changed.
- Undo and redo any number of changes made previously to the graphic.
- Create groups of elements to bind them together.
- Create a path graphic from multiple open line elements.

### Drawing and Dragging Elements

You can create elements such as lines, curves, circles, squares, and so on. You can combine these elements to create complex drawings of all the equipment in your manufacturing environment.

After you draw an element, you can modify its properties. For more information about modifying properties, see *Editing Element Properties* on page 61.

Regardless of the kind of element you are drawing, drawing each kind of element is very similar.

After you draw an element, the pointer tool is selected again by default. To draw multiple elements of the same type, double-click the element in the Tools panel. It remains selected after you draw your first element of that type. You can press the ESC key to return to the pointer tool again.

If you draw or drag an element outside of the visible canvas area to the right or bottom, horizontal and/or vertical scroll bars appear but the visible area does not follow the mouse. You can later use the scroll bars to scroll the canvas and see the element you drew or moved.

## Drawing Rectangles, Rounded Rectangles, Ellipses, and Lines

You can draw rectangles, rounded rectangles, ellipses, and lines on the canvas.

### To draw a rectangle, rounded rectangle, ellipse, or line

1. Click the appropriate icon in the **Tools** panel.
2. Click the canvas and drag the shape of the element on the canvas.
3. When you are done, release the mouse button.

## Drawing Polylines, Polygons, Curves, and Closed Curves

You can draw polylines, polygons, curves, and closed curves on the canvas.

If you are drawing a closed element, the element automatically closes when you are done drawing.

### To draw a polyline, polygon, curve, or closed curve

1. Click the appropriate icon in the **Tools** panel.
2. Click the canvas where you want to start the element.
3. Click the next point for the element.
4. Continue clicking until you have all the points you require.
5. When you are done, right-click.
6. You can change the shape of these elements anytime by editing their control points. For more information, see *Editing Control Points* on page 123.

## Drawing 2-Point Arcs, 2-Point Pies and 2-Point Chords

You can draw 2-point arcs, 2-point pies, and 2-point chords on the canvas.

If you are drawing a closed element, the element automatically closes when you are done drawing.

### To draw a 2-point arc, 2-point pie, or 2-point chord

1. Click the appropriate icon in the **Tools** panel.
2. Click the canvas where you want to start the element and hold the mouse button.
3. Drag the mouse to where you want the element to end.
4. When you are done, release the mouse button.
5. You can change the shape of these elements anytime by editing their control points. For more information, see *Editing Control Points* on page 123.

## Drawing 3-Point Arcs, 3-Point Pies, and 3-Point Chords

You can draw 3-point arcs, 3-point pies and 3-point chords on the canvas.

If you are drawing a closed element, the element automatically closes when you are done drawing.

### To draw a 3-point arc, 3-point pie, or 3-point chord

1. Click the appropriate icon in the **Tools** panel.
2. Click the canvas where you want to start the element.

3. Click the canvas in two other places to define the element.
4. You can change the shape of these elements anytime by editing their control points. For more information, see *Editing Control Points* on page 123.

## Placing and Importing Images

You can place an image element on the canvas and import an image into it.

### To draw an image

1. Click the image icon in the **Tools** panel.
2. Click the canvas and drag the shape of the image element.
3. Release the mouse button. The **Open** dialog box appears.
4. Browse to the image file, select it, and then click **Open**. The image file is loaded into the image element.

## Drawing Buttons

You can draw a button on the canvas. You can configure a button with a text label or an image.

For more information on how to configure a button with an image after drawing it on the canvas, see *Configuring Buttons with Images* on page 122.

### To draw a button

1. Click the button icon in the **Tools** panel.
2. Click the canvas and drag the shape of the button element.
3. Release the mouse button. The button text appears in edit mode.
4. Type a text label for the button and click **Enter**.

## Placing Text

You can place text on the canvas.

The text element has no border and no background fill. The text does not wrap. When you type the text, the size of the Text element expands.

You can also drag the handles of the Text element to resize it.

### To place text

1. Click the text icon in the **Tools** panel.
2. Click the canvas where you want to place the text.
3. Type the single line of text you want.
4. When you are done, do one of the following:
  - Click Enter to type a new line of text. This new line is a new element.
  - Click the canvas outside the text element.

## Drawing Text Boxes

You can draw text boxes on the canvas. Text boxes can have borders and background fill.

You can also configure the text to wrap in the text box. For more information, see *Wrapping Text in Buttons* on page 122.

**To draw a text box**

1. Click the text box icon in the **Tools** panel.
2. Click the canvas where you want to place the text box.
3. Drag a rectangle on the canvas.
4. Release the mouse button. The text appears in edit mode.
5. Type a text label for the text box, and then click **Enter**.

## Drawing Status Elements

Use the status element to indicate specific quality and status conditions of attributes.

**To draw status elements**

1. Click the status icon in the **Tools** panel.
2. Click the canvas where you want to place the status element.
3. Drag a rectangle on the canvas.
4. Release the mouse button.

## Drawing User Interface Common Controls

Draw user interface common controls on the canvas to add additional functionality to your graphic. Each of the controls has specific behavior when it is drawn. For example, you can change the width of a combo box, but not the height.

**To draw a windows control**

1. Click the appropriate common interface control icon in the **Tools** panel.
2. Click the canvas where you want to place the control.
3. Drag a rectangle on the canvas.
4. Release the mouse button.

## Dragging Elements

After you draw elements on the canvas, you can drag them to a new position.

**To drag elements on the canvas**

1. Select one or more elements.
2. Click one of them and hold the mouse button down.
3. Drag the mouse to the new position.
4. Release the mouse button.

## Editing Element Properties

You can control the appearance of an element, a group of elements, or multiple elements with functions on the toolbar and properties shown in the **Properties Editor** of the Industrial Graphic Editor.

Properties	Toolbox	A
<b>Graphic</b>		
Name	<b>Ellipse1</b>	
<b>Appearance</b>		
ElementStyle	None	
X	<b>990</b>	
Y	<b>450</b>	
Width	<b>330</b>	
Height	<b>220</b>	
Angle	<b>1</b>	
AbsoluteOrigin	<b>1155, 560</b>	
RelativeOrigin	<b>0, 0</b>	
Transparency	0	
Locked	False	
<b>Fill Style</b>		
FillOrientation	RelativeToGra	
FillColor		<b>Solid</b>
UnfilledColor		<b>Solid</b>
FillBehavior	Both	
HorizontalDirection	Right	
HorizontalPercentFi	100	
VerticalDirection	Up	
VerticalPercentFill	100	
<b>Line Style</b>		
LineWeight	1	
LinePattern	Solid	
LineColor		<b>Solid</b>
<b>Runtime Behavior</b>		
Enabled	True	
TabOrder	0	
TabStop	True	
Visible	True	

Often you can edit an element by changing the values of its properties instead of using the mouse to perform the same function. This is useful when you want very exact editing, such as when you want to resize an element to a specific width.

The **Properties Editor** shows the properties common to all selected elements.

- Read-only properties appear in grey.
- Non-default values appear in bold.

**Note:** The **Properties Editor** not only supports values, but also allows input of color, font, and file information in the respective dialog boxes.

Properties are organized in categories so you can find them more easily. The following table shows the categories:

Property Category	Purpose
<b>Graphic</b>	Element name or other describing identifiers

---

<b>Property Category</b>	<b>Purpose</b>
<b>Appearance</b>	Element style, location, size, orientation, offset, transparency and locked status
<b>Fill Style</b>	Any parameters related to the fill appearance of the element
<b>Line Style</b>	Any parameters related to the appearance of element lines
<b>Text Style</b>	Any parameters related to the appearance of element text
<b>Runtime Behavior</b>	Element visibility, tab order and any other element behavior at run time
<b>Custom Properties</b>	Additional user-defined properties you can associate with any element

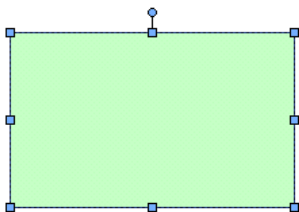
---

## Selecting Elements

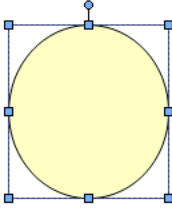
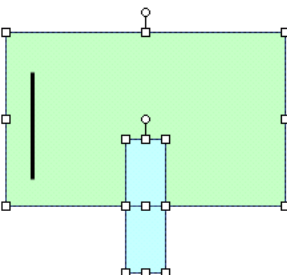
You can select one or more graphic elements from the Industrial Graphic Editor canvas by:

- Clicking on them with the mouse.
- Dragging a lasso around them with your mouse.
- Selecting them with a menu option or with a shortcut key.
- Selecting them by name from the **Elements** list.

When you select an element, handles appear around the border of the element that can be moved to control the size of the element. You can change the orientation of a graphic element by moving the handle connected to the top center border handle.



When you select multiple elements, the last selected element is the primary element. All other previously selected elements are secondary elements.

Selected Element	Description
<p data-bbox="250 283 454 315">Primary Element</p> 	<p data-bbox="592 283 974 315">Appears with color-filled handles.</p> <p data-bbox="592 325 1055 357">Behaves as an active selected element.</p> <p data-bbox="592 367 1136 462">Is the point of reference for all operations, such as aligning or spacing multiple selected elements.</p>
<p data-bbox="250 588 503 619">Secondary Elements</p> 	<p data-bbox="592 588 893 619">Appear with white handles.</p> <p data-bbox="592 630 1039 661">Behave as inactive selected elements.</p> <p data-bbox="592 672 1120 703">Follow the edits made to the primary element.</p>

To select a group, you must click one of the elements contained in the group.

## Selecting Elements by Mouse Click

You can select one or more elements by pressing Shift + clicking. This is particularly useful for selecting multiple elements that are not necessarily all included in a specified rectangular area on the canvas.

### To select an element or multiple elements by mouse click

1. On the canvas, click an element. It becomes selected.
2. To select further elements, press Shift+ click. The other elements become selected.

**Note:** You can see in the Elements List which elements are selected.

## Selecting Elements by Lasso

You can select one or more elements by lassoing them with your mouse. This is useful for selecting multiple elements within a specified rectangular area on the canvas.

### To select elements by lasso

1. On the canvas, click outside any element and hold the mouse button down.
2. Drag the mouse so that the lasso wraps around all elements that you want to select.
3. When you are done, release the mouse button. The elements that are fully enclosed within the lasso are selected.

## Selecting All Elements

You can select all elements using the Select All function.

### To select all elements

- On the **Edit** menu, click **Select All**. All elements on the canvas are selected.

---

**Note:** You can also press the F2 key to select all elements.

---

## Selecting Elements Using the Elements List

You can use the Elements List to select any elements on the canvas. The Elements List is particularly useful for selecting elements behind other elements.

The Elements List shows which elements are currently selected. The primary selected element appears by default in dark blue, the secondary selected elements appear by default in light blue.

---

**Note:** The color setting of the Elements List depends on the setting for the **Selected Items** option in the operating system's **Display Properties Appearance** panel.

---

### To select elements using the Elements List

1. In the **Elements List**, select the element name.
2. To select multiple elements, Ctrl + click the other elements.

## Unselecting Elements

You can unselect one or more selected elements. You can do this by clicking on them individually on the canvas or in the Elements List.

If you want to remove the selected elements in a specified rectangular area, you can use the lasso.

### To unselect elements individually

1. Do one of the following:
  - Shift + click the selected element on the canvas.
  - Ctrl + click the selected element name in the Elements List.
2. Repeat the previous step for all elements you want to unselect.

### To unselect elements from a specified rectangular area

1. Shift + click the canvas outside of any element.
2. Drag the mouse so that the lasso surrounds the elements that you want to unselect.
3. Release the mouse button. The selected elements within the lasso are unselected, and the selected elements outside the lasso remain selected.

## Inline Editing

After you place graphic elements on the Industrial Graphic Editor canvas, you can edit them by selecting them and clicking on them again. This is called inline editing. The following elements support inline editing.:

---

Element	Use inline editing to
Button, text, text box	Edit the text.
Polyline, polygon, curve, closed curve	Edit the control points.
2-point arc, 2-point pie, 2-point chord, 3-point arc, 3-point pie, 3-point chord	Edit the start and sweep angles.

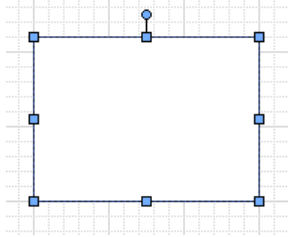
---



Element	Use inline editing to
Group	Edit the individual elements and groups contained in the group.
Path	Edit the control points.

### To edit elements with inline editing

1. Select an element by clicking on it or selecting it from the **Elements** list. Element handles appear around the border of the element.



2. Click the element again to begin inline editing.
  - For buttons, text, and text boxes, the text is selected and you can type new text.
  - For polylines, polygons, curves, and closed curves, the control points of the element appear. Move a control point to change the shape of the element.

You can also add and delete control points. For more information, see *Adding and Removing Control Points* on page 123.

- For arcs, pies, and chords, the handles for the start angle and sweep angle appear. Move a handle to change the start angle and sweep angle.
  - For groups, the group handle is replaced with a shaded outline. You can select individual elements and groups within the group to edit and move them.
3. Click the canvas outside the element to exit from inline editing.

## Copying, Cutting, and Pasting Elements

After you draw elements, you have the same cut, copy, and paste options available to you as in any other Windows application. However, some of these options behave differently in the Industrial Graphic Editor.

You can also duplicate elements. Duplicating elements lets you quickly make copies of existing selected elements without first copying or cutting. You can duplicate one or more selected elements at the same time.

When you copy or duplicate elements, all of its properties are copied with the element. If you do not want the properties to be identical, you must change the properties after you copy.

Locked grouped elements and the path element behave differently when you copy or duplicate them.

If you copy or duplicate:

- A set of elements that are locked, the copy is not locked.
- Grouped elements, the copy is still grouped.
- A path element, the copy is also a path.

## Copying Elements

After you select an element, you can copy it by using menu options or you can Ctrl + click.

### To copy one or more elements

Do any of the following:

- Select one or more elements to be copied on the canvas. On the **Edit** menu, click **Copy**. On the **Edit** menu, click **Paste**. The paste pointer appears. Click the canvas where you want to place the copy.
- Ctrl + click an element.
- Select one or more elements to be copied on the canvas. Press Ctrl + C. Press Ctrl + V. The paste pointer appears. Click the canvas where you want to place the copy.

## Cutting or Deleting Elements

You can cut elements or groups or you can delete them. Cutting lets you select elements or groups and remove them from the canvas. You can paste the removed elements or groups.

Deleting elements or groups deletes them from the canvas. You cannot paste deleted elements or groups.

### To cut one or more elements

- Select one or more elements, and then do one of the following:
  - On the **Edit** menu, click **Cut**.
  - Press Ctrl + X.

### To cut and paste elements on the canvas

1. Select the element or group.
2. On the **Edit** menu, click **Cut**.
3. Do one of the following:
  - Click **Paste** on the **Edit** menu.
  - Press Ctrl + V.
4. Click the canvas location where you want the element or group to be placed.

### To delete an element or a group

1. To remove the element or group and **not** use it in the future, select the element or group.
2. Do one of the following:
  - Click **Delete** on the **Edit** menu.
  - Press Delete on your keyboard.

## Duplicating Elements

Duplicating elements enables you to select an element or elements and quickly make copies of them.

You can also specify the amount of overlap when you duplicate.

### To duplicate elements

1. Select one or more elements.
2. Do one of the following:
  - a. Click **Duplicate** on the **Edit** menu. The selected element is duplicated and appears offset to the original element.
  - b. Press Ctrl + D. The selected element is duplicated and appears offset to the original element.

- c. Ctrl + click one of the selected elements to duplicate all selected elements. You can keep the mouse button down and drag them to the new position on the canvas.

### To set the overlap when you duplicate

1. Duplicate an element or elements. The element is copied overlapping the original.
2. Move the duplicated element to the location relative to the original. For example, move the duplicated element five grid spaces above the original element.
3. Duplicate the element again. The new duplicate is placed in the same offset you specified in the preceding step. For example, five grid spaces above the original element.

## Moving Elements

After you create elements, you can move them to the location you want on the canvas.

You can move elements or groups by dragging them to the new location or you can open the properties for the element or group and change the X and Y properties.

If you turned on snap to grid, moving an element or group with the mouse snaps the element or group to the grid.

If you move an element or group by specifying X and Y coordinates, it does not snap to the grid.

You can move an element or group vertically or horizontally using the keyboard.

### To move an element or group using the mouse

1. Select the element or group you want to move.
2. Drag the elements or group to the new location.

### To move an element or group by specifying the X and Y properties

1. Select the element or group you want to move.
2. In the Properties Editor, expand **Appearance**.
3. Do the following:
  - In the **X** box, type the new X location.
  - In the **Y** box, type the new Y location.
4. Click in the canvas or click ENTER.

### To move an element or group vertically or horizontally using the mouse

1. Shift + click to select the element or group you want to move.
2. Drag the elements or group to the new location.

### To move an element or group vertically or horizontally using the keyboard

1. Select the element or group you want to move.
2. Do one of the following:
  - Press the Up or Down arrow keys to move the element or group vertically by one unit in the grid.
  - Press the Left or Right arrow keys to move the element or group horizontally by one unit in the grid.

---

**Note:** You can move the element or group by two units in the grid by additionally pressing the Shift key, by four units by additionally pressing the Ctrl key, and by 10 units by additionally pressing both keys.

---

### To move multiple elements or groups

1. Select the elements and/or groups.

2. Move them as you would with one single element. The elements are moved together and maintain their spatial relationship when moving.

## Aligning Elements

After you draw elements, you can align them:

- Horizontally so that their top or bottom sides or their center points are horizontally aligned.
- Vertically so that their left, right, or center points are vertically aligned.
- So that their center points are on top of each other.
- So that their points of origin are on top of each other.

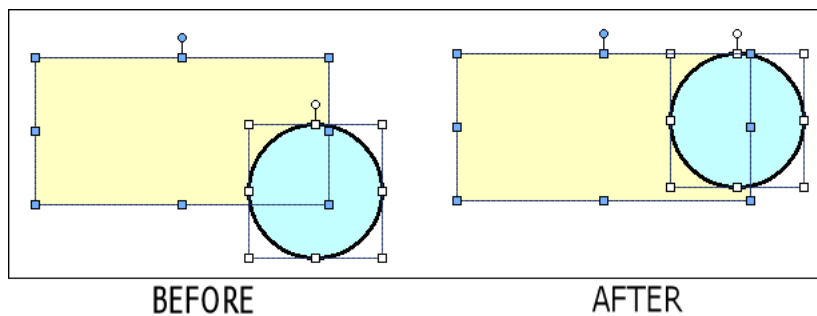
When you align elements, the secondary elements are moved so that they align with the primary element. For more information about primary and secondary elements, see *Selecting Elements* on page 62.

## Aligning Elements Horizontally

You can align multiple elements by their top or bottom sides or horizontally on their middle points.

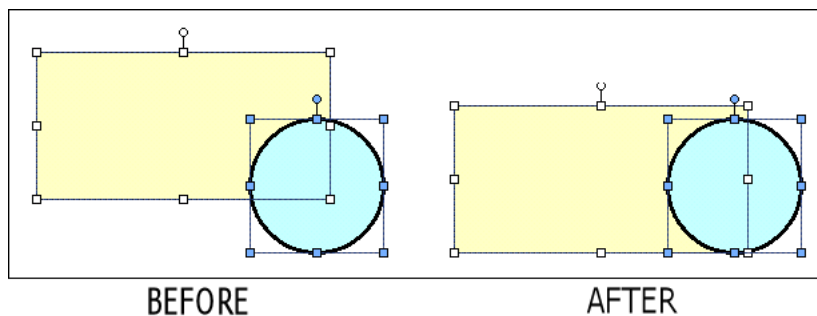
### To align elements by their top sides

1. Select all elements that you want to align. Make sure the element you want to align all other elements to is the primary element.
2. On the **Arrange** menu, point to **Align**, and then click **Align Top**. The secondary elements are moved so that their top sides are aligned with the top side of the primary element.



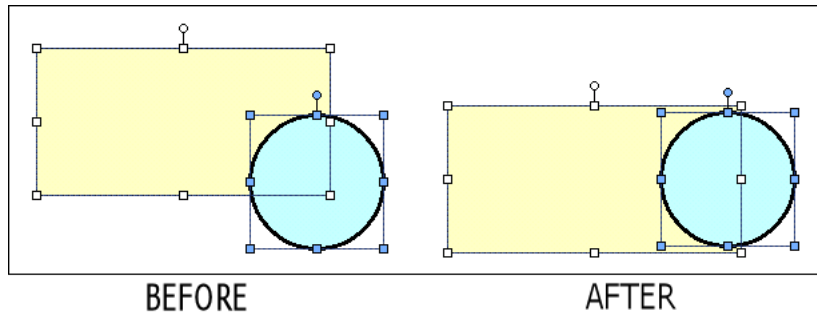
### To align elements by their bottom sides

1. Select all elements that you want to align. Make sure the element you want to align all other elements to is the primary element.
2. On the **Arrange** menu, point to **Align**, and then click **Align Bottom**. The secondary elements are moved so that their bottom sides are aligned with the bottom side of the primary element.



### To align elements horizontally by their center points

1. Select all elements that you want to align. Make sure the element you want to align all other elements to is the primary element.
2. On the **Arrange** menu, point to **Align**, and then click **Align Middle**. The secondary elements are moved vertically so that their center points are aligned with the center point of the primary element.

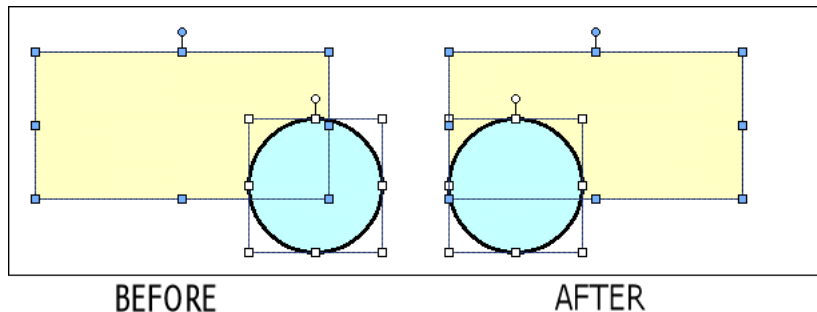


## Aligning Elements Vertically

You can vertically align multiple elements on the left, right, or their center points.

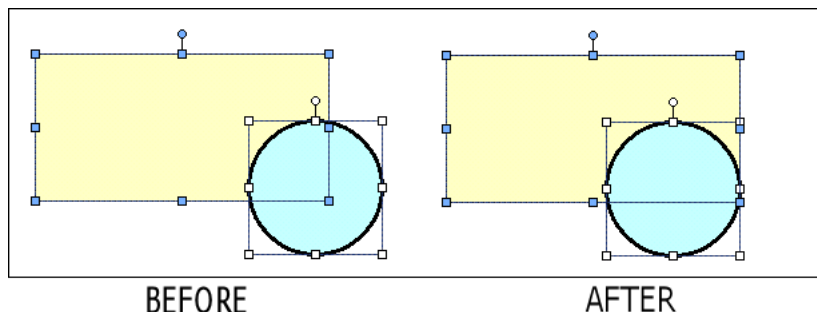
### To align elements by their left sides

1. Select all elements that you want to align. Make sure the element you want to align all other elements to is the primary element.
2. On the **Arrange** menu, point to **Align**, and then click **Align Left**. The secondary elements are moved so that their left sides are aligned with the left side of the primary element.



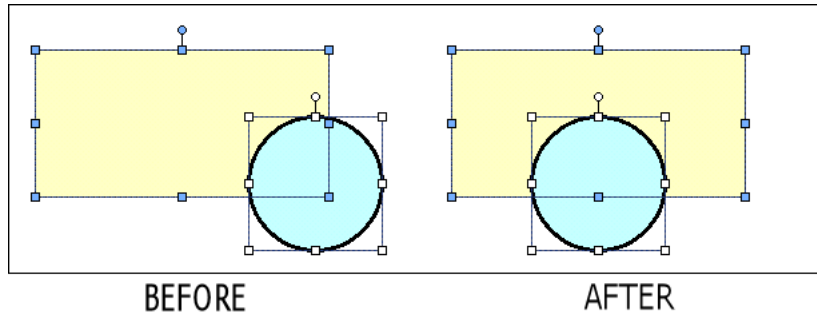
### To align elements by their right sides

1. Select all elements that you want to align. Make sure the element you want to align all other elements to is the primary element.
2. On the **Arrange** menu, point to **Align**, and then click **Align Right**. The secondary elements are moved so that their right sides are aligned with the right side of the primary element.



### To align elements vertically by their centers

1. Select all elements that you want to align. Make sure the element you want to align all other elements to is the primary element.
2. On the **Arrange** menu, point to **Align**, and then click **Align Center**. The secondary elements are moved horizontally so that their center points are aligned with the center point of the primary element.

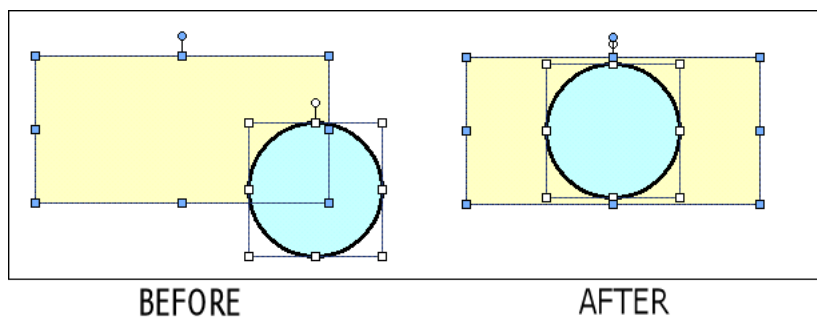


## Aligning Elements by their Center Points

You can align elements by their center points. The center point of one or more elements is the point halfway between the horizontal and vertical boundaries.

### To align elements on their center points

1. Select all elements that you want to align. Make sure the element you want to align all other elements to is the primary element.
2. On the **Arrange** menu, point to **Align**, and then click **Align Centers**. The secondary elements are moved so that their center points are placed on top of the center point of the primary element.



## Aligning Elements by their Points of Origin

You can align elements by their points of origin. By default, the element's center point is the point of origin. But, an element's center point can be changed. The center point is the anchor point of an element to the canvas.

### To align elements on their points of origin

1. Select all elements that you want to align. Make sure the element you want to align all other elements to is the primary element.
2. On the **Arrange** menu, point to **Align**, and then click **Align Origins**. The secondary elements are moved so that their points of origins are placed on top of the point of origin of the primary element.

## Adjusting the Spacing between Elements

You can adjust the space between elements according to specific rules.

You can adjust the spacing between elements in the following ways:

- Horizontally - moves the selected elements left or right without changing the vertical positions.
- Vertically - moves the selected elements up or down without changing the horizontal positions.
- Distribution - moves the selected elements so that their center points are distributed in equal distance to each other.
- Equal spacing - moves the selected elements so that the distance between their edges is equal.
- Increase spacing - moves all selected elements one pixel further away from each other. The primary element does not move.
- Decrease spacing - moves all selected elements one pixel closer toward each other. The primary element does not move.
- Remove spacing - removes all space between selected elements so that their edges touch.

## Distributing Elements

You can distribute elements so that their center points are distributed in equal distance to each other.

### To distribute elements horizontally

1. Select at least three elements.
2. On the **Arrange** menu, point to **Space**, and then click **Distribute Horizontal**. The selected elements are distributed horizontally.

### To distribute elements vertically

1. Select at least three elements.
2. On the **Arrange** menu, point to **Space**, and then click **Distribute Vertical**. The selected elements are distributed vertically.

## Making Space between Elements Equal

You can space elements so that the distances between their boundaries are equal.

The difference between making space between elements equal and distributing them is that making space equal uses the boundaries of the elements, whereas distributing uses the center points. Both do not necessarily lead to the same result.

### To make the horizontal space between elements equal

1. Select at least three elements.
2. On the **Arrange** menu, point to **Space**, and then click **Make Horizontal Space Equal**. The selected elements are moved so that the horizontal spaces between their boundaries are equal.

### To make the vertical space between elements equal

1. Select at least three elements.
2. On the **Arrange** menu, point to **Space**, and then click **Make Vertical Space Equal**. The selected elements are moved so that the vertical spaces between their boundaries are equal.

## Increasing Space between Elements

You can increase space between elements equally.

The primary element does not move. All secondary elements are moved away from the primary element.

### To increase the horizontal space between elements

1. Select at least two elements.
2. On the **Arrange** menu, point to **Space**, and then click **Increase Horizontal Spacing**. The selected elements are moved so that the horizontal space between them is increased by one pixel.
3. Repeat the previous step to move the selected elements further away from each other.

### To increase the vertical space between elements

1. Select at least two elements.
2. On the **Arrange** menu, point to **Space**, and then click **Increase Vertical Spacing**. The selected elements are moved so that the vertical space between them is increased by one pixel.
3. Repeat the previous step to move the selected elements further away from each other.

## Decreasing Space between Elements

You can decrease space between elements equally.

The primary element does not move. All secondary elements move toward the primary element. You can move them until the left sides of all elements are aligned.

### To decrease the horizontal space between elements

1. Select at least two elements.
2. On the **Arrange** menu, point to **Space**, and then click **Decrease Horizontal Spacing**. The selected elements are moved so that the horizontal space between them is decreased by one pixel.
3. Repeat the previous step to move the selected elements closer toward each other.

### To decrease the vertical space between elements

1. Select at least two elements.
2. On the **Arrange** menu, point to **Space**, and then click **Decrease Vertical Spacing**. The selected elements are moved so that the vertical space between them is decreased by one pixel.
3. Repeat the previous step to move the selected elements closer toward each other.

## Removing All Space between Elements

You can remove all space between selected elements so that their boundaries touch.

The primary element does not move. All secondary elements move toward the primary element. You can move them until the left and right sides of all secondary elements are aligned.

### To remove all horizontal space between elements

1. Select all elements between which you want to remove the space.
2. On the **Arrange** menu, point to **Space**, and then click **Remove Horizontal Spacing**. The horizontal space between all selected elements is removed, so that their boundaries touch.

### To remove all vertical space between elements

1. Select all elements between which you want to remove the space.
2. On the **Arrange** menu, point to **Space**, and then click **Remove Vertical Spacing**. The vertical space between all selected elements is removed, so that their boundaries touch.

## Resizing Elements

You can resize selected elements by:



- Dragging the handles of a single element to increase or decrease its horizontal or vertical size.
- Changing the Width and Height properties of one or more elements using the Properties Editor.
- Proportionally resizing multiple elements.
- Making multiple objects the same width and/or height.

Some elements cannot be resized or can only be resized in certain directions, such as the Calendar control or DateTime Picker. If the primary element has such restrictions, then any secondary elements resize proportional to the change in primary element's size and do not resize independently.

## Resizing a Single Element with the Mouse

You can resize a single selected element with the mouse.

You can resize most elements to any given width and height, or to a fixed width to height ratio.

### To resize a single selected element with the mouse

1. Select an element. The handles of the selected element appear.
2. Drag one of the handles. The object is resized while you drag.
3. Release the mouse button.

### To resize a single selected element with the mouse and keeping a fixed width/height ratio

1. Select an element. The handles of the selected element appear.
2. Press and hold the Shift key.
3. Drag one of the handles. The object is resized while you drag, the width/height ratio stays unchanged.
4. Release the mouse button and Shift key.

## Resizing Elements by Changing Size Properties

You can resize one or more elements by changing the width and/or height property of the selected elements.

### To resize elements by changing their size properties

1. Select one or more elements.
2. In the Properties Editor, type a value for **Width** and for **Height**. The selected elements are resized accordingly.

## Resizing Elements Proportionally

You can resize multiple elements proportionally on the canvas. One element is the primary element you can use to resize. The secondary elements resize proportionally to the change of the primary element.

### To resize elements proportionally

1. Select multiple elements.
2. Drag one of the handles of the primary element. The secondary elements are resized accordingly by the same percentage.
3. Release the mouse button.

For example, assume the primary element is 100 pixels wide and 50 pixels high. A secondary element is 200 pixels wide and 20 pixels high.

You drag the handle of the primary element so that it is 120 pixels wide (20 percent increase) and 100 pixels high (100 percent increase).

Then the secondary element is resized to 240 pixels wide (20 percent increase of the original width of 200 pixels) and 40 pixels high (100 percent increase of the original width of 20 pixels).

## Making Elements the Same Width, Height, or Size

You can make elements the same width, height, or size.

### To make elements the same width

1. Select at least two elements. Make sure the primary element is the element with the target width for all elements.
2. On the **Arrange** menu, point to **Size**, and then click **Make Same Width**. The width of the secondary elements are resized to the same width as the primary element.

### To make elements the same height

1. Select at least two elements. Make sure the primary element is the element with the target height for all elements.
2. On the **Arrange** menu, point to **Size**, and then click **Make Same Height**. The height of the secondary elements are resized to the same height as the primary element.

### To make elements the same size

1. Select at least two elements. Make sure the primary element is the element with the target size for all elements.
2. On the **Arrange** menu, point to **Size**, and then click **Make Same Size**. The size of the secondary elements are resized to the same size as the primary element.

## Adjusting the z-Order of Elements

The z-order of elements specifies which element appears on top of other elements when the elements overlap on the canvas. The z-order also determines how the elements of a path graphic connect.

When you place new elements on the canvas, they are placed at the top and can cover all other elements.

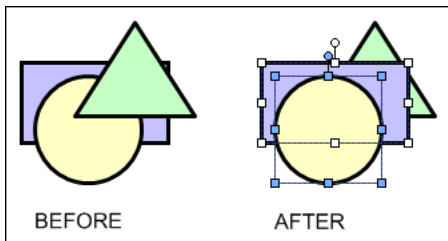
However, you might want to bring certain elements forward so that they are always visible or overlap certain other elements. Or you may want to use a large background element behind all other elements. You can:

- Bring one or more elements to the very front.
- Send one or more elements to the very back.
- Bring one or more elements one level forward.
- Send one or more elements one level backward.

You can use the Elements List to see or change the z-order of the elements.

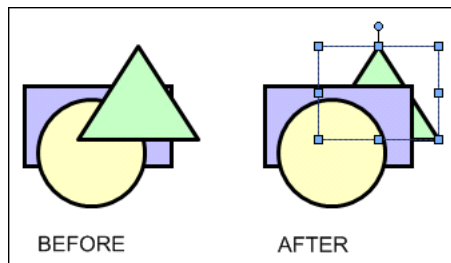
### To bring selected elements to the front

- On the **Arrange** menu, point to **Order**, and then click **Bring To Front**. The selected elements are brought to the front. They do not change their relative z-order.



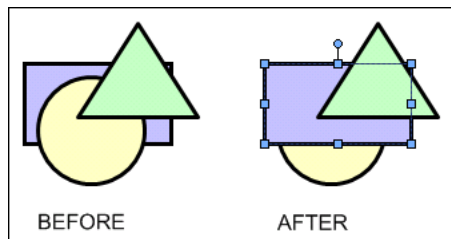
### To send selected elements to the back

- On the **Arrange** menu, point to **Order**, and then click **Send To Back**. The selected elements are sent to the back. They do not change their relative z-order.



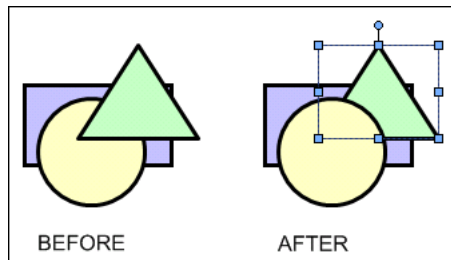
### To bring selected elements one level forward

- On the **Arrange** menu, point to **Order**, and then click **Bring Forward**.



### To send selected elements one level backward

- On the **Arrange** menu, point to **Order**, and then click **Bring Backward**.



## Rotating Elements

You can rotate elements to any orientation (0 - 359 degrees):

- Graphically with the rotation handle.
- Numerically by typing the orientation angle in the Properties Editor.

- By rotating in 90 degree increments in a clockwise or counter-clockwise direction.

The element rotates around its point of origin. By default, the point of origin is in the center of the element. You can move the point of origin to any other location, even outside of the object itself. To change the point of origin, see *Moving the Origin of an Element* on page 77.

## Rotating Elements with the Mouse

You can rotate one or more elements with the mouse. If you select multiple elements, you can rotate the primary element. The secondary elements rotate in unison with the primary element.

You can rotate elements:

- Freely in the range 0 to 359 in integer degrees.
- In multiples of 15 degrees.
- In multiples of 45 degrees.

You can rotate an element with the rotation handle. The rotation handle is a light-blue circle at the top of a selected element.

### To rotate elements freely with the mouse

1. Select one or more elements.
2. Grab the rotation handle of the primary element.
3. Drag the mouse across the screen. All selected elements are rotated around their own points of origin as you move the mouse.
4. Release the mouse button.

### To rotate elements by multiple of 15 degrees with the mouse

1. Select one or more elements.
2. Grab the rotation handle of the primary element.
3. Press and hold the Shift key.
4. Drag the mouse across the screen. All selected elements are rotated in multiples of 15 degrees around their own points or origin as you move the mouse.
5. Release the mouse button and Shift key.

### To rotate elements by multiple of 45 degrees with the mouse

1. Select one or more elements.
2. Grab the rotation handle of the primary element.
3. Press and hold the Ctrl key.
4. Drag the mouse across the screen. All selected elements are rotated in multiples of 45 degrees around their own points or origin as you move the mouse.
5. Release the mouse button and Ctrl key.

## Rotating Elements by Changing the Angle Property

You can change the angle property of one or more selected elements.

### To rotate elements by changing the angle property

1. Select one or more elements.
2. In the Properties Editor, type a value in the **Angle** box.

3. Click Enter. The selected elements rotate to the specified angle.

## Rotating Elements by 90 Degrees

You can rotate elements in 90 degrees clockwise or counter-clockwise increments.

To rotate elements by multiples of 15 and 45 degrees, see *Rotating Elements with the Mouse* on page 76.

### To rotate elements by 90 degrees clockwise

1. Select one or more elements.
2. On the **Arrange** menu, point to **Transform**, and then click **Rotate Clockwise**. The selected elements rotate by 90 degrees clockwise.

### To rotate elements by 90 degrees counter-clockwise

1. Select one or more elements.
2. On the **Arrange** menu, point to **Transform**, and then click **Rotate Counter Clockwise**. The selected elements rotate by 90 degrees counter-clockwise.

## Moving the Origin of an Element

You can change the point of origin of any element. The point of origin specifies around which point the element rotates or flips. By default the point of origin is in the center of the element.

You can change the point of origin:

- With the mouse on the canvas.
- By specifying the absolute origin in the Properties Editor.
- By specifying the relative origin in the Properties Editor.

## Changing Points of Origin with the Mouse

You can change the point of origin for an element with the mouse.

### To change the point of origin for an element with the mouse

1. Select an element on the canvas.
2. Move the mouse over the rotation handle of the element. The point of origin icon for the element appears.
3. Drag the Point of Origin icon to where you want to place the new point of origin for the element.
4. Release the mouse button.

## Changing Points of Origin in the Properties Editor

You can change the absolute or relative point of origin in the Properties Editor.

The absolute point of origin shows the position of the point of origin in relation to the canvas. The absolute point of origin changes when the element moves.

The relative point of origin shows the position of the point of origin in relation to the center of the element. The relative point of origin does not change when the element moves.

### To change the point of origin in the Properties Editor

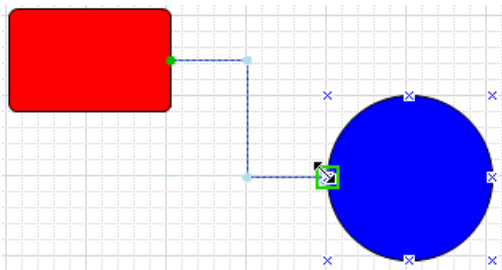
1. Select one or more elements on the canvas.
2. In the Properties Editor, do one of the following:

- Type the absolute coordinates in the x, y format for the point of origin.
  - Type the relative coordinates in the x, y format for the point of origin.
3. Click Enter. The points of origin move to the specified absolute position or to the specified position in relation to the center points of the selected elements.

For example, if you have two elements, you can set the relative point of origin to 10, 10 to place the points of origin for both elements 10 pixels to the right and 10 pixels below the corresponding center points of each element.

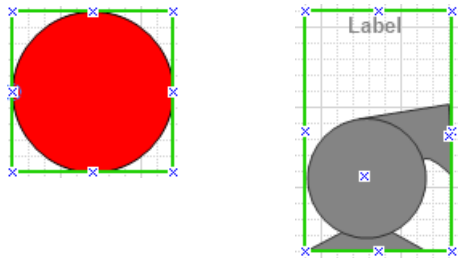
## Add Connectors Between Graphic Elements

A connector is a line drawn between graphic elements. A connector starts at a connection point on one graphic element and ends at a connection point on another element. Connectors are particularly useful for complex graphics like flow diagrams, industrial piping, or electrical wiring diagrams that incorporate many lines between graphic elements.

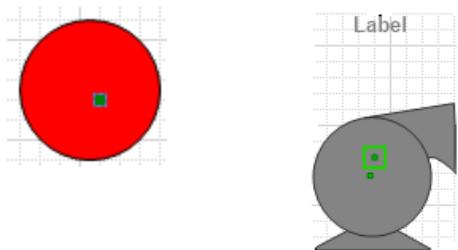


Connectors change length or orientation in response to changes to connected graphic elements during design time or run time. Graphic elements within a graphic can be moved, resized, or rotated and still maintain the connector between elements.

Connection points are the locations on a graphic element to attach a connector. A default set of eight connection points appear on the bounding rectangle around a graphic element or an embedded graphic.



You can also add custom connection points to graphic elements or embedded graphics if you want to place a connector at a different position than on a bounding rectangle.



One or more control points appear on an Angled connector based on the number of angles in the connector path. Using your mouse, you can move a control point horizontally or vertically to change the shape of a connector between the fixed connection points on both graphic elements. By default, a control point is placed at the intersection point of each right angle in a connector.

---

**Important:** Connector lines do not maintain their horizontal and vertical orientation with 90 degree angles when placed in a graphic whose dimensions exceed 1280 by 1280 pixels. Instead, the connector will revert to a straight line between connection points.

---

You can also add control points to a connector if you want to change the shape of its path. For more information about adding control points to a connector, see *Change the Shape of a Connector* on page 83.

## Draw a Connector

Use the Connector tool to draw a connector between graphic elements. The Connector tool initially attempts to draw a connector with a minimum number of angles. You can change the shape of the initial connector path using control points to redraw the path if necessary.

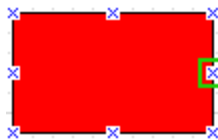
A connector supports Symbol Wizards like any other graphic element. You can associate a connector with a Symbol Wizard layer by dragging the connector element to the layer during design time. You can also remove the connector from a layer by removing the connector from the association list. If a connector is hidden based on the Symbol Wizard's Wizard Option configuration, the connector does not appear during run time.

Press the Esc key to cancel drawing a connector. Also, clicking on the Industrial Graphic Editor's canvas takes you out of connector drawing mode.

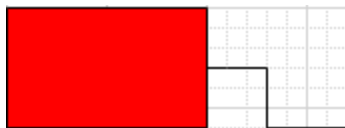
### To draw a connector

1. Open a graphic in Industrial Graphic Editor that you want to add a connector.
2. Click once on the **Connector** icon within the **Tools** pane to draw a single connector.  
If you want to draw multiple connectors, double click on the **Connector** icon.
3. Move your mouse over the first graphic element that you want to add a connector.  
The default connection points appear when you move your mouse over a graphic element.
4. Place the mouse over the connection point where you want to place the connector on the graphic element.

A green rectangle appears around the connection point when it is selected.

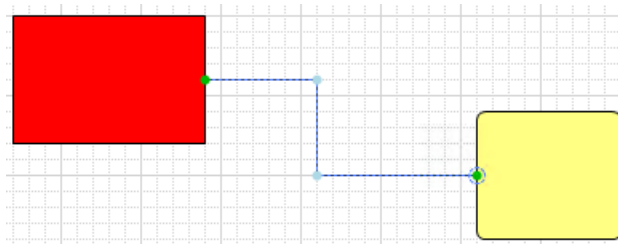


A start connection point has an automatic offset, which is a perpendicular straight line segment from the start connection point to the first angle in the connector path. An automatic offset prevents the connector from following the border of the originating graphic element. No automatic offset is applied to the terminating connection point on the connected graphic element.



5. Press and hold your left mouse key and drag the mouse to the second graphic element.

Connection points appear when you move the mouse over the second graphic element.



6. Release the mouse button when you are over a selected connection point on the second graphic element.

The connector appears as a line between both graphic elements.

7. If necessary, use connector control points to change the shape of the connector between the connected graphic elements.

## Adding Connection Points

Use the Connection Point tool to place additional connection points at other locations on a graphic element than the bounding rectangle. Also, you can place custom connection points on an embedded graphic and connect to them.

---

**Note:** Custom connection points are part of the parent graphic element and cannot be grouped. Also, custom connection points added to a graphic element that is part of a Symbol Wizard layer are shown when the graphic element is part of the Symbol Wizard's current configuration.

---

Press the Esc key to cancel adding a connection point. Also, clicking on the Industrial Graphic Editor's canvas takes you out of connection point addition mode.

### To add connection points

1. Open a graphic that you want to add one or more connection points.
2. Click once on the **Connection Point** icon from the **Tools** pane to draw a single connection point on a graphic element.

If you want to draw multiple connection points, double click on the **Connection Point** icon.

3. Move your mouse to a location within a graphic element that you want to place a new connection point.

---

**Note:** Connection points can be added within the bounding rectangle of a hosting element of an embedded graphic.

---

4. Click once.

The new connection point appears as a green rectangle at the location you selected.

### To change the position of a connection point

You can change the position of a connection point that you added to a graphic element or a graphic.

1. Click on the connection point you want to move to select it.
2. Keep the left mouse key pressed.
3. Drag the connection point to a new location and release the mouse key.

The **X** and **Y** properties show the coordinate position of the connection point.

You can also change the location of a connection point you added by changing the X and Y coordinate values assigned to the connection point's **X** and **Y** properties.



## Change Connector Properties

A connector includes a set of **Appearance**, **Line Style**, and **Runtime Behavior** properties. These properties can be modified during design time.

During run time, you can use animation to change property values that affect the appearance or behavior of a connector. For Angled or Straight connectors, you can use Line Style or Element Style animation. A connection point does not support any type of animation.

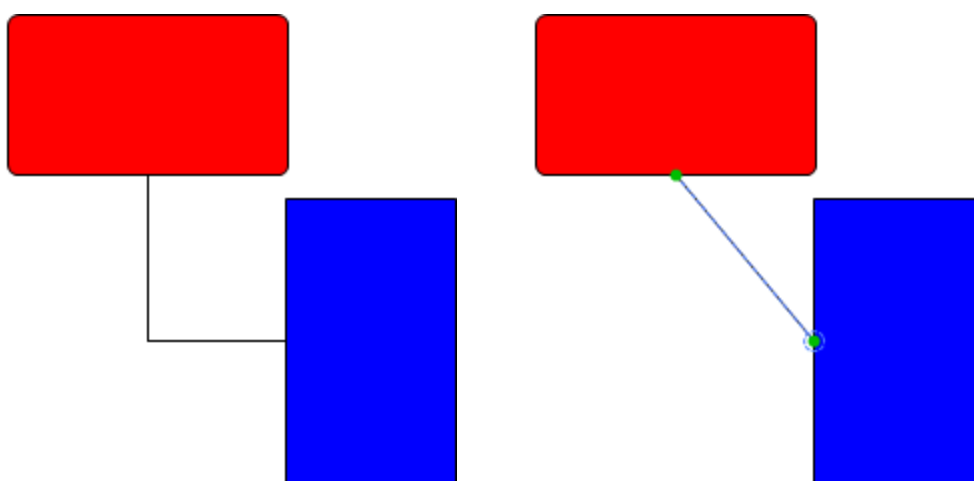
Property	Description
<b>Appearance Properties</b>	
ConnectionType	Type of connector (Angled or Straight). Angled is the default, which contains a set of connector line segments with 90 degree angles between them.
ElementStyle	Element style applied to a connector to change the line color, fill, and pattern. Line styles can be applied to Angled and Straight types of connectors. None is the default.
X	Horizontal coordinate of the left most border of a selected graphic element or graphic that has an attached connector. The X coordinate value is the number of pixels between the left vertical border of the Industrial Graphic Editor's canvas area to the left most border position of the selected graphic element or graphic.
Y	Vertical coordinate of the top border of a selected graphic element or graphic that has an attached connector. The Y coordinate value is the number of pixels between the top horizontal border of the Industrial Graphic Editor's canvas area to the top of the selected graphic element or graphic.
Start	Read-only X and Y coordinates of a connector's start point with respect to the origin at the top left corner of the Industrial Graphic Editor's canvas.
End	Read-only X and Y coordinates of a connector's end point with respect to the origin at the top left corner of the Industrial Graphic Editor's canvas.
<b>Line Style Properties</b>	
LineWeight	Line weight of an Angled or Straight type of connector. 1 is the default.
LinePattern	Line pattern of an Angled or Straight type of connector. Solid is the default.
StartCap	Shape of the line start point of an Angled or Straight type of connector. Flat is the default.
EndCap	Shape of line end point of an Angled or Straight type of connector. Flat is the default.
LineColor	Line color of an Angled or Straight type of connector. Black is the default.
<b>Runtime Behavior Properties</b>	
Enabled	Connector animation is enabled or disabled during run time. Enabled is the default.

Property	Description
Visible	Connector is visible or hidden during run time. Visible is the default.

## Change the Type of Connector

You can select the type of connector by setting an option for the **ConnectionType** property in the **Appearance** pane of the Industrial Graphic Editor.

The default connector type is Angled, which consists of horizontal and vertical lines with a 90 degree angle between them. A Straight connector is a straight line between the connection points on different graphic elements.



### To change the type of connector

1. Click on a connector to select it.

The **Connection Type** property appears in the **Appearance** pane of the Industrial Graphic Editor.

2. Select a connector type from the drop-down list of the **Connection** property.

The appearance of the selected connector changes to the type you selected.

## Change the Length of a Connector

You can change the length of a connector to move it to another connection point on a graphic element or detach it from a graphic element.

---

**Note:** A connector is not required to start or end at a connection point on a graphic element. Connectors can be drawn on the canvas detached from any graphic elements.

---

### To change the length of a connector

1. Click on a connector to select it.

The start point of a connector appears as a green circle. A halo appears around the end point.

2. Select either the start or end point of the connector and keep your left mouse key pressed.
3. Drag the start or end point of a connector to a new location and release the mouse key.

The length of the connector changes until you release your mouse key. The **Start** or **End** properties show a new coordinate position based on whether you moved the connector's start or end point.

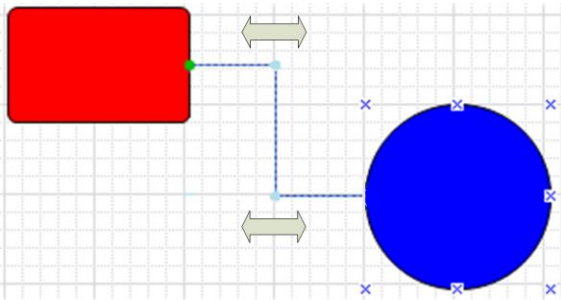
You can also change the location of a connector's start or end points by changing the X and Y coordinate values assigned to the connector's **Start** or **End** properties.

## Change the Shape of a Connector

An Angled connector includes one or more control points that can be moved to change the shape of a connector.

The movement of control points is restricted to changing the shape of the connector without changing the fixed position of the connection points on graphic elements. For example, the two control points shown in the figure below are part of an Angled connector. Both control points can be moved horizontally to the same X coordinate position to change the position of the vertical line segment of the connector.

But, the line segments from the connection points of the graphic elements to their adjacent control points cannot be moved. In the following example, the control points cannot be moved vertically. To maintain the required right angles between line segments of an Angled connector would require the locations of the fixed connection points to be moved.



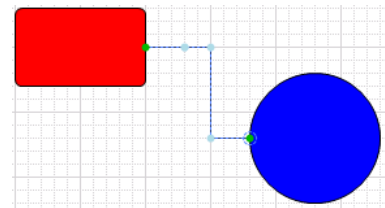
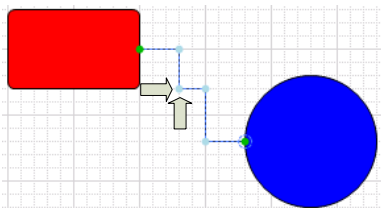
### To change the shape of a connector

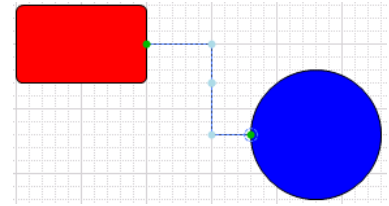
1. Click on a connector to select it.  
A control point appears at the intersection point of each right angle in an Angled connector.
2. Click a control point and keep your mouse key pressed.  
The mouse cursor changes to a double arrow to indicate the control point can be moved.
3. Move the control point to change the shape of the connector and release the mouse key.

### Delete a Control Point

If you want to remove a control point to change the shape of your connector, you must first reposition a line segment to ensure the resulting connector path contains only right angles before deleting a control point.

In the following example, a line segment needs to be repositioned vertically or horizontally to ensure the connector contains only right angles. After a line segment is repositioned, a control point can be deleted.





### To delete a control point

1. Click on a connector to select it.
2. Move a line segment within the connector to ensure the connector path contains only right angles between its line segments.
3. With your Ctrl key pressed, place your cursor over the control point on the connector you want to delete.

The appearance of the cursor changes to a pen tip with a minus sign to indicate that a control point can be deleted from a connector.

---

**Note:** Control points at the right angles of a connector cannot be deleted. You can only remove control points on straight line segments.

---

4. Left click with your mouse to delete the control point.

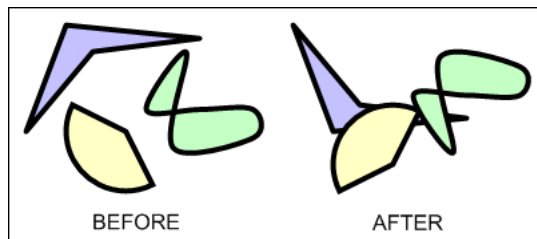
The small blue circle on the connector disappears indicating the control point is deleted from the connector.

## Flipping Elements

You can flip elements on their horizontal or vertical axes. The axis for each element is determined by its point of origin. For more information on how to change the point of origin, see *Moving the Origin of an Element* on page 77.

### To flip elements vertically

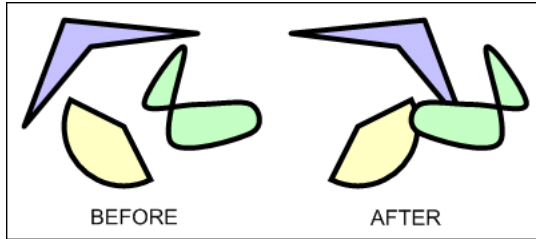
1. Select one or more elements.
2. On the **Arrange** menu, point to **Transform**, and then click **Flip Vertical**. The selected elements are flipped vertically on their horizontal axis.



### To flip elements horizontally

1. Select one or more elements.

2. On the **Arrange** menu, point to **Transform**, and then click **Flip Horizontal**. The selected elements are flipped horizontally on their vertical axis.



## Locking and Unlocking Elements

When you lock elements, they cannot be:

- Moved.
- Resized.
- Rotated.
- Aligned.
- Flipped.

You also cannot change the point of origin in locked elements. To enable these functions again, you must unlock the elements.

### To lock elements

1. Select all elements that you want to lock.
2. Do one of the following:
  - On the **Arrange** menu, click **Lock**.
  - In the Properties Editor, set the Locked property to **True**.

The selected elements appear with lock icons at their handles.

### To unlock elements

1. Select all elements that you want to unlock.
2. Do one of the following:
  - On the **Arrange** menu, click **Unlock**.
  - In the Properties Editor, set the Locked property to False.

The lock icons disappear from the handles of the selected elements.

## Making Changes Using Undo and Redo

Use the Undo function to reverse an editing change you made to a graphic element in the Industrial Graphic Editor. After you undo a change, you can also restore the change by using the Redo function. The Undo and Redo functions appear as icons on the Industrial Graphic Editor's **Edit** menu.



You can undo one single change, or any number of changes that you have previously made. You can also redo any number of changes. These can be selected from a list.

### To undo a single change

- Do one of the following:

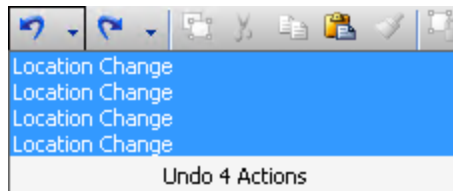
- Press Ctrl + Z.
- On the **Edit** menu, click **Undo**.

### To redo a single change

- Do one of the following:
  - Press Ctrl + Y.
  - On the **Edit** menu, click **Redo**.

### To undo a specified number of previous changes

1. On the toolbar, click the **Undo** icon. The **Undo** list appears with one or more descriptions of what changes were made.



2. Select a change from the list or click the bottom list entry to undo all changes. The changes up to and including the selected item are undone.

### To redo a specified number of previously undone changes

1. On the toolbar, click the **Redo** icon. The **Redo** list appears with a description of what the undone changes were.
2. Select a the change from the list or click the bottom list entry to redo all changes. The changes down to and including the selected item are redone.

## Working with Groups of Elements

You can group together multiple elements. This is useful to bind certain element together so that they are not inadvertently moved. The group is treated as a new element.

You can:

- Create a group from one or more elements.
- Ungroup the elements without losing their original configuration information.
- Add more elements to an existing group.
- Remove elements from a group.
- Edit the elements of a group without having to ungroup them.

## Creating a Group of Elements

After you create elements, you can group them. Grouping elements lets you manage the elements as one unit.

Groups are assigned default names when you create them, such as Group1, Group2, and so on. After you create a group, you can rename it.

Groups can have properties that are different than the properties of the elements.

### To create a group

1. Select the elements you want as part of the new group.

2. On the **Arrange** menu, point to **Grouping**, and then click **Group**. The elements are combined into a group. The group is listed in the Elements List.
3. Rename the group as required. To do this:
  - a. In the Elements List, click the group name and click again. The group name is in edit mode.
  - b. Type a new name and click Enter. The group is renamed.
  - c. You can also rename a group or elements by changing the Name property in the Properties Editor.

## Ungrouping

After you create a group, you can ungroup it if you no longer want it.

If the group included elements and other groups, when you ungroup, the original elements and groups again exist as independent items. To ungroup any subgroups, you must select each one and ungroup it separately.

If you ungroup a set of elements and elements already exist with the names of the grouped elements, then the newly ungrouped elements are renamed.

### To ungroup

1. Select the groups you want to ungroup.
2. On the **Arrange** menu, point to **Grouping**, and then click **Ungroup**. The groups is converted to the original elements. The group name is removed from the Elements List and the element names appear.

## Adding Elements to Existing Groups

After you create a group, you can add elements or other groups to an existing group.

For example, you can combine a group that represents a valve with another group that represents a tank to create a new group that can be called a tank unit.

You can add:

- Elements to groups.
- Groups to the primary selected group.

### To add elements to an existing group

- On the canvas, select the group and also elements and groups that you want to add.
- Right-click a selected element or on the group, point to **Grouping**, and then click **Add to Group**. The selected elements are added to the group.

---

**Note:** You can also add elements to existing groups by using the Elements List in similar way.

---

## Removing Elements from Groups

After you create a group, you can remove elements from the group. This lets you remove one or more elements you no longer want in that group.

Removing elements from the group removes them from the canvas. It also removes any scripts or animations you added to the element.

### To remove an element from a group

1. On the canvas, select the group with the elements that you want to remove.
2. Click the group again to enter inline editing mode.

3. Select the elements that you want to remove from the group.
4. Right-click a selected elements, point to **Grouping**, and then click **Remove from Group**. The selected elements are removed from the group.

---

**Note:** You can also remove elements from existing groups by using the Elements List in similar way.

---

## Editing Components within a Group

You can edit components within a group without having to dissolve the group. Do this by:

- Selecting the element in Elements List.
- Using the **Edit Group** command on the shortcut menu.
- Slowly double-clicking to enter inline editing mode.

### To edit components within a group by using the Elements List

1. In the Elements List, expand the group that contains the element that you want to edit.
2. Select the element that you want to edit. The element appears selected in the group and the group is outlined with a diagonal pattern.
3. Edit the element with the Properties Editor, by mouse or by menu according to your requirements.
4. Click outside the group.

### To edit components within a group by using the Edit Group command

1. On the canvas, select the group that you want to edit.
2. On the menu **Edit**, click **Edit Group 'GroupName'**. The group is outlined with a diagonal pattern.
3. Select the element that you want to edit.
4. Edit the element with the Properties Editor, by mouse, by menu or pop-up menu according to your requirements.
5. Click outside the group.

---

**Note:** If you move the position of an element in a group outside the group, the group size is automatically changed to incorporate the new position of the element.

---

## Using Path Graphics

You can join a set of open elements, such as lines, to create a new closed element. The new closed element is called a path graphic.

You can:

- Create a path graphic by joining open elements.
- Break the path graphic into its elements.
- Edit the path graphic in its entirety or by editing its elements.
- Add new elements to the path graphic.
- Remove elements from the path graphic.

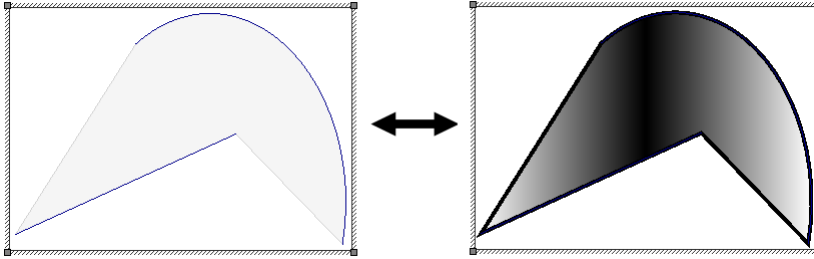
You can view a path graphic in two modes:

- Element mode shows you the individual elements contained in the path graphic and determine its shape. Elements that make up the path graphic are shown as blue lines. The points where the elements are connected are shown as grey lines.



- Path mode shows you the path graphic in its final rendering, including fill styles and lines styles.

When you are in inline editing mode, you can switch between both modes by pressing the space bar. This lets you preview the path graphic without leaving the inline editing mode.

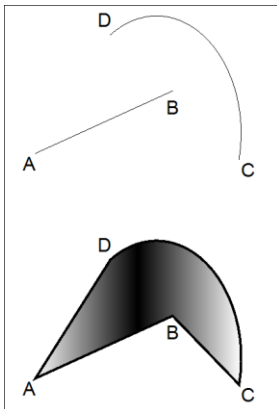


## Creating a Path Graphic

You can create a path graphic from one or more open elements such as lines, polylines, curves, and arcs.

The path graphic is created according to the start and end points and the z-order of the open elements that create it.

For example, if you draw a line from point A to point B, then an arc from point C to point D, and then join these elements in a path graphic, the path graphic is described by a straight edge from points A to B, a straight edge from points B to C, a curved edge from points C to D, and closed by a straight edge from points D to A.



**Note:** If the Path Graphic doesn't appear as you expected after you create it, then you can swap the end points or change the z-order of one or more elements. For more information, see *Swapping the End Points of an Element in a Path Graphic* on page 91 and *Changing the Z-order of an Element in a Path Graphic* on page 92.

### To create a path graphic

1. Select one or more open elements.
2. On the **Arrange** menu, point to **Path**, and then click **Combine**. A new path graphic is created from the selected open elements.

## Breaking the Path of a Path Graphic

You can break the path of a path graphic so that it is broken into its individual open elements. When you do so, the path graphic loses its unique properties such as fill style and line style.

### To break the path of a path graphic

1. Select one or more path graphics.
2. On the **Arrange** menu, point to **Path**, and then click **Break**.

## Changing a Path Graphic

You can edit an existing path graphic on the canvas by accessing the individual elements of which it consists. For each individual element, you can:

- Move.
- Rotate.
- Change size.
- Change start and sweep angles if the elements are arcs.
- Change control points if the elements are curves or polylines.
- Swap the end points of an element in a path graphic.
- Change the z-order or the elements in a path graphic.

The path graphic is updated while you edit the individual elements.

## Moving Elements in a Path Graphic

You can move elements in a path graphic. If you move an element outside of the path graphic boundary, the boundary is redrawn to include the moved element.

### To move an element within a path graphic

1. Select the path graphic you want to edit.
2. Do one of the following:
  - a. On the **Edit** menu, click **Edit Path**.
  - b. Slowly double-click the path graphic.The path graphic appears in element mode.
3. Select the individual element within the path graphic you want to move. You can also do this by selecting the element in the Elements List.
4. Click a solid part of the element and drag it to the new position. The element is moved.
5. Click outside the path graphic on the canvas. The path graphic is shown in path mode.

## Resizing Elements in a Path Graphic

You can resize elements in a path graphics. If you resize an element outside of the path graphic boundary, the boundary is redrawn to include the resized element.

### To resize an element within a path graphic

1. Select the path graphic you want to edit.
2. Do one of the following:
  - a. On the **Edit** menu, click **Edit Path**.
  - b. Slowly double-click the path graphic.The path graphic appears in element mode.

3. Select the individual element within the path graphic you want to resize. You can also do this by selecting the element in the Elements List.
4. Click and drag any of the resize handles of the selected element. The element is resized.
5. Click outside the path graphic on the canvas. The path graphic is shown in path mode.

## Editing Start and Sweep Angles of Elements in a Path Graphic

If your path graphic contains arcs, you can edit the start and sweep angles of these elements. If changing the angle of an element causes it to overlap the path graphic boundary, the boundary is redrawn to include the changed element.

### To edit start or sweep angle of an element within a path graphic

1. Select the path graphic you want to edit.
2. Do one of the following:
  - a. On the **Edit** menu, click **Edit Path**.
  - b. Slowly double-click the path graphic.The path graphic appears in element mode.
3. Select the individual element within the path graphic for which you want to change the start or sweep angle. You can also do this by selecting the element in the Elements List.
4. Click the element again. The element appears in edit mode with its start angle and sweep angle.
5. Click outside the path graphic on the canvas. The path graphic is shown in path mode.

## Editing Element Control Points in a Path Graphic

If your path graphic contains curves or polylines, you can edit the control points of these elements. If changing the control points of the element causes it to overlap the path graphic boundary, the boundary is redrawn to include the changed element.

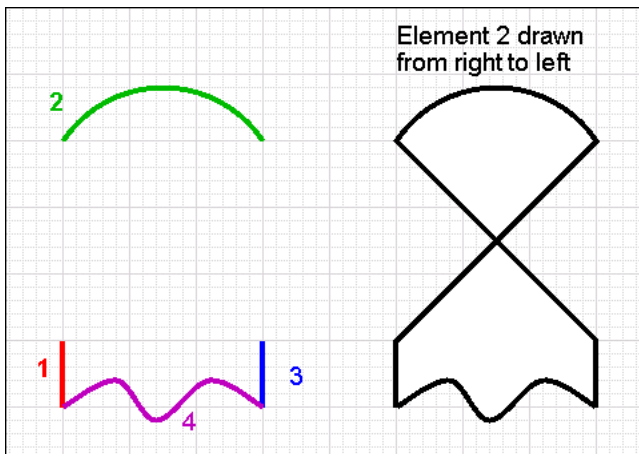
### To edit control points of an element within a path graphic

1. Select the path graphic you want to edit.
2. Do one of the following:
  - a. On the **Edit** menu, click **Edit Path**.
  - b. Slowly double-click the path graphic.The path graphic appears in element mode.
3. Select the curve or polyline element within the path graphic for which you want to change the control points. You can also do this by selecting the element in the Elements List.
4. Click the element again. The element appears in edit mode with its control points.
5. Drag any of the control points to shape the curve or polyline.
6. Click outside the path graphic on the canvas. The path graphic is shown in path mode.

## Swapping the End Points of an Element in a Path Graphic

The path graphic is created by following the direction in which you draw its elements.

If a path graphic does not appear as expected, this can be caused by drawing an element in a different direction as intended. You can see this if one of the path graphic edges appears crossed over when connecting to the previous and next element.



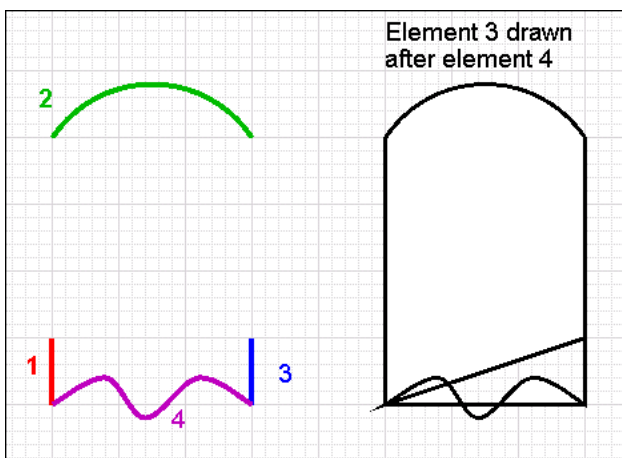
You can fix this by swapping the end points of the element where this appears.

### To swap the end points of an element within a path graphic

1. Select the path graphic you want to edit.
2. Do one of the following:
  - a. On the **Edit** menu, click **Edit Path**.
  - b. Slowly double-click the path graphic.
 The path graphic appears in element mode.
3. Select the individual element within the path graphic for which you want to swap the end points. You can also do this by selecting the element in the Elements List.
4. Right-click that element and select **Path, Swap End Points** on the context menu. The end points of the selected element are swapped and the path graphic is updated accordingly.
5. Click outside the path graphic on the canvas. The path graphic is shown in path mode.

### Changing the Z-order of an Element in a Path Graphic

If a path graphic does not appear as expected, this can be caused by drawing an element in a different z-order as intended. You can see this if one of the path graphic edges jumps across the path graphic area.



You can fix this by changing the z-order of the element where this appears.

---

**Note:** The z-order of elements in a path graphic is only applicable within the path graphic.

---

### To change the z-order of an element within a path graphic

1. Select the path graphic you want to edit.
2. Do one of the following:
  - a. On the **Edit** menu, click **Edit Path**.
  - b. Slowly double-click the path graphic.The path graphic appears in element mode.
3. Select the individual element within the path graphic for which you want to change the z-order. You can also do this by selecting the element in the Elements List.

---

**Note:** You can see the elements in their z-order in the Elements List. Alternatively, you can select one from the Elements List and change its z-order.

---

4. On the **Arrange** menu, point to **Order**, and then click:
  - **Send To Back** to send the element to the back of the set of elements of the path graphic.
  - **Send Backward** to send the element one order backward.
  - **Sent To Front** to send the element to the front of the set of elements of the path graphic.
  - **Send Forward** to send the element one order forward.
5. Click outside the path graphic on the canvas. The path graphic is shown in path mode.

## Adding Elements to an Existing Path Graphic

You can easily add elements to an existing path graphic. You can add:

- New elements, which you draw while the path graphic is in edit mode.
- Existing elements, which are already on the canvas.

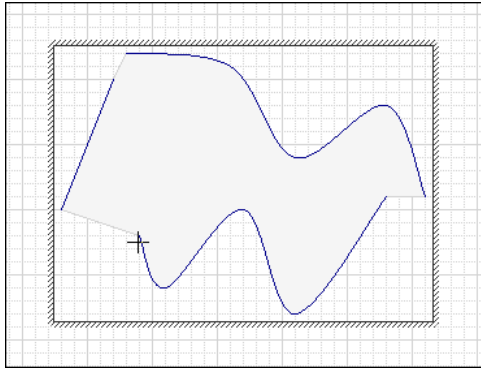
You can only add open elements such as lines, polylines, curves, and arcs to an existing path graphic.

You can only set the origin of a new element within the frame of the existing path graphic. If you click anywhere outside the path graphic, the edit mode is exited and the element you are drawing is a new element.

### To add new elements to an existing path graphic

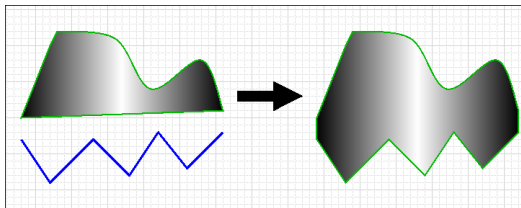
1. Select the path graphic to which you want to add a new element.
2. On the **Edit** menu, click **Edit Path**. The path graphic appears in element mode.
3. Select the new element you want to add from the Tools panel.

4. Draw the element as you would normally. While you are drawing the element, the path graphic is updated.



### To add existing elements to an existing path graphic

1. Select the path graphic and all elements that you want to add to the path graphic.
2. Right-click a solid part of a selected element, point to **Path**, and then click **Add To Path**. The selected elements are added to the selected path graphic.



## Removing Elements from a Path Graphic

You can remove individual elements from a path graphic. The elements are not deleted, but appear outside the path graphic.

You cannot remove the last element of a path graphic.

### To remove elements from a path graphic

1. Select the path graphic from which you want to delete individual elements.
2. On the **Edit** menu, click **Edit Path**. The path graphic appears in element mode.
3. Shift + click one or more elements to remove.

---

**Note:** You can also select the elements to remove from the Elements List by holding CTRL key during the selection.

---

4. Right-click any selected element, point to **Path**, and then click **Remove From Path**. The selected element is removed from the path graphic and the path graphic is updated accordingly.

# CHAPTER 5

## Editing Common Properties of Elements and Graphics

### Editing the Name of an Element

Some properties are common to most types of elements, such as fill, line styles, and visibility. You can:

- Edit the name of an element.
- Edit the fill properties of an element.
- Edit the line properties of an element.
- Edit the text properties of an element.
- Set the style.
- Set the transparency level of an element.
- Adjusting colors and style for an element's gradient style.
- Enable and disable elements for run-time interaction.
- Change the visibility of an element.
- Change the tab order of an element.
- Use the Format Painter to format elements.
- Edit the general properties of a graphic.

The name of an element uniquely identifies the element on the drawing surface.

When you draw a new element on the drawing surface, it is assigned a default name. You can then change its name in the Properties Editor or the Elements List.

Element names are case-insensitive and unique within the same element hierarchy. It is possible to have two elements with the same name if one is, for example, in a group and the other outside that group.

#### **To change an element's name in the Properties Editor**

1. Select the element on the drawing surface.
2. In the Properties Editor, click the value for the **Name** box.
3. Type a new name and click Enter.

#### **To change an element's name in the Elements List**

1. Select the element in the Elements List.
2. Click the element in the Elements List again.
3. Type a new name and click Enter.

### Editing the Fill Properties of an Element

You can configure the following fill properties for an element:

- Fill style as solid color, gradient, pattern or texture
- Unfilled style
- Fill orientation, relative to the element or to the screen
- Fill behavior, which determines if the object is to be filled horizontally, vertically, or both
- Horizontal fill direction
- Vertical fill direction
- Percent of horizontal fill
- Percent of vertical fill

## Setting Fill Style

You can configure the fill style of one or more elements. You can do this to:

- Selected elements on the toolbar.
- Style properties in the Properties Editor.
- Nested style properties, such as just one color of a multi-colored gradient.

### To configure the fill style of an element with the toolbar

1. Select one or more elements you want to configure.
2. On the toolbar, click the down arrow to the right of the **Fill Color** icon. The fill style list appears.
3. Configure the fill color. Do any of the following:
  - Click **No Fill** to configure an empty element.
  - Click a predefined solid color in the display.
  - Click **More Solid Colors** to open the style selection dialog box and select a solid color.
  - Click **Color Picker** to select a color from the screen.
4. Configure the fill gradient, pattern, or texture. Do any of the following:
  - Click a predefined gradient.
  - Click **More Gradients** to open the style selection dialog box and configure a gradient.
  - Click **Patterns** to open the style selection dialog box and select a pattern.
  - Click **Textures** to open the style selection dialog box and select a texture.

For more information about the style selection dialog box, see *Setting Style* on page 103.

### To configure the fill style by setting style properties

1. Select one or more elements.
2. In the Properties Editor, locate the FillStyle property.
3. Click the browse button to open the style selection dialog box. For more information about the style selection dialog box, see *Setting Style* on page 103.

### To configure the fill style by setting gradient color style properties

1. Select one or more elements with gradient fill style.
2. In the Properties Editor, locate the **Color1**, **Color2**, and **Color3** properties.
3. Click the browse button for any of these to set the selected gradient color from the style selection dialog box. For more information, see *Setting Style* on page 103.



## Setting Unfilled Style

You can configure an element's unfilled style. The unfilled style of an element determines the element's unfilled portion at design time and run time.

### To configure the unfilled style of an element

1. Select one or more elements.
2. In the Properties Editor, click **UnfilledStyle**.
3. Click the browse button in the **UnfilledStyle** line. The style selection dialog box appears.
4. Select a solid color, gradient, pattern, or texture. For more information about the style selection dialog box, see *Setting Style* on page 103.
5. Click **OK**.

## Setting Fill Orientation

You can configure an element's fill orientation in the Properties Editor. The fill orientation property determines if the fill style is relative to the screen or element.

- If relative to the screen, the gradient, pattern, or texture does not rotate with the element.
- If relative to the element, the gradient, pattern, or texture rotates with the element.

### To configure an element's fill orientation

1. Select one or more elements you want to configure.
2. In the Properties Editor, click **FillOrientation**.
3. From the list in the same line, click **RelativeToScreen** or **RelativeToGraphic**.

## Setting Fill Behavior

You can set the fill behavior of an element. The fill can be:

- Horizontal.
- Vertical.
- Both horizontal and vertical.

### To set an element's fill behavior

1. Select one or more elements you want to configure.
2. In the Properties Editor, set the property **FillBehavior** to one of the following:
  - **Horizontal**
  - **Vertical**
  - **Both**

## Setting Horizontal Fill Direction and Percentage

An element can fill:

- From left to right.
- From right to left.

You can also set the amount you want the element to be horizontally filled by as a percentage.

**To set an element's horizontal fill direction and percentage**

1. Select one or more elements you want to configure.
2. In the Properties Editor, set the HorizontalDirection property to:
  - **Right** to fill from left to right.
  - **Left** to fill from right to left.
3. For the HorizontalPercentFill property, type a percentage (0 - 100) in the value box.

## Setting Vertical Fill Direction and Percentage

An element can fill:

- From bottom to top.
- From top to bottom.

You can also set the amount you want the element to be vertically filled by as a percentage.

**To set an element's vertical fill direction and percentage**

1. Select one or more elements you want to configure.
2. In the Properties Editor, set the VerticalDirection property to:
  - **Top** to fill from bottom to top.
  - **Bottom** to fill from top to bottom.
3. For the **VerticalPercentFill** property, type a percentage (0 - 100) in the value box.

## Editing the Line Properties of an Element

You can set the line properties for any element that contains lines, such as:

- Lines and polylines.
- Rectangles, rounded rectangles, and ellipses.
- Curves, closed curves, and polygons.
- Arcs, pies, and chords.
- Text boxes.

You can set the:

- Start and end points for lines, arcs, and H/V lines.
- Line weight, which is the thickness of a line.
- Line pattern, which is the continuity of a line. For example, a continuous line, a dotted line, a dashed line, or a combination.
- Line style, which is the fill style of a line.
- Shape and size of the end points of a line. For more information, see *Setting Line End Shape and Size* on page 118.

---

**Note:** You can also set the element's line properties in the **Line Format** properties group in the Properties Editor.

---

## Setting Start or End Points of a Line

After you draw a line or H/V line, you can change its start or end points in the Properties Editor.

**To set the line or H/V line start or end point**

1. Select a line or H/V line.
2. In the Properties Editor, type coordinate values X, Y for the **Start** or **End** properties.

## Setting the Line Weight

You can set a line weight from 0 pixels to 255 pixels for any element that contains lines. You can set the line weight using the **Format** menu, the toolbar, or the LineWeight property in the Properties Editor.

---

**Note:** Large line weight settings can cause unexpected behavior, especially with curves and line end styles.

---

**To set the line weight using the Format menu**

1. Select one or more elements.
2. On the **Format** menu, click **Line Weight**.
3. To use a predefined line weight, select it from the list.
4. To use another line weight, click **More Line Options**. The **Select Line Options** dialog box appears. In the **Weight** box, type a new line weight from 0 to 255 and then click **OK**.

## Setting the Line Pattern

You can set the line pattern for any element that contains lines. The line pattern specifies the continuity of a line (continuous, dotted, dashed) and not its fill properties.

**To set the line pattern**

1. Select one or more elements.
2. On the **Format** menu, click **Line Pattern**.
3. To use a predefined line pattern, select it from the list.
4. To use another line pattern, click **More Line Options**. The **Select Line Options** dialog box appears. In the **Pattern** list, select a pattern, and then click **OK**.

---

**Note:** You can also set the line pattern by changing the **LinePattern** property in the Properties Editor.

---

## Setting the Line Style

You can set the line style for any element that contains lines. Setting the line style is similar to setting the fill style. You can also set the solid color, gradient, pattern, and texture for a line.

**To set the line style**

1. Select one or more elements.
2. On the toolbar, click the **Line Color** icon. The line style list appears.
3. Configure the line color. Do any of the following:
  - Click a predefined solid color in the display.
  - Click **More Solid Colors** to open the style selection dialog box and select a solid color.
  - Click **Color Picker** to select a color from the screen.
4. Configure the line gradient, pattern, or texture. Do any of the following:
  - Click a predefined gradient.
  - Click **More Gradients** to open the style selection dialog box and configure a gradient.

- Click **Patterns** to open the style selection dialog box and select a pattern.
- Click **Textures** to open the style selection dialog box and select a texture.

For more information about the style selection dialog box, see *Setting Style* on page 103.

---

**Note:** You can also set the element's line style in the Properties Editor. If you do this, you can configure the solid color, gradient, pattern, or texture in the style selection dialog box. For more information, see *Setting Style* on page 103.

---

## Setting the Text Properties of an Element

You can set the following for text, text box, and button elements:

- The text that appears
- The format in which the text appears
- The font of the text
- The alignment of the text
- The text style

You can also substitute strings in text, text box, and button elements.

## Setting the Displayed Text

You can set the text of a text element, text box, or button in the canvas or by changing the Text property in the Properties Editor.

### To set the text to display

1. Select the text element, text box or button on the canvas.
2. On the **Edit** menu, click **Edit Text**. The selected element appears in edit mode.
3. Type a text string and press **Enter**.

## Setting the Text Display Format

You can configure how values are shown for the text in a text box or button. For example, as a rounded float with the format `#.###`.

You can format the text display for the:

- Text element and the button element in the same way as in the HMI or with the `TextFormat` property in the Properties Editor.
- Text box element only with the `TextFormat` property.

### To set the text display format

1. Select a text element, text box, or button.
2. In the **Properties Editor**, type a format for the `TextFormat` property.

## Setting the Text Font

You can change the font style and font size of a text using:

- The **Format** menu.
- The **Font** property in the **Properties Editor**.
- Lists on the toolbar.

**To set the text font, font style, and size**

1. Select a text element, a text box, or a button element on the canvas.
2. On the **Format** menu, click **Fonts**. The **Font** dialog box appears.
3. Set the font, font style, size, and effects.
4. Click **OK**.

## Setting the Text Color

You can set the text color as a solid color, a gradient, a pattern, or a texture.

---

**Note:** You can also change the text color in the **Properties Editor** with the **TextColor** property.

---

**To set the text color**

1. Select a text element, a text box, or a button element on the canvas.
2. Click the **Text Color** icon.
3. Configure the text color. Do any of the following:
  - Click a predefined solid color in the display.
  - Click **More Solid Colors** to open the style selection dialog box and select a solid color.
  - Click **Color Picker** to select a color from the screen.
4. Configure the text gradient, pattern, or texture. Do any of the following:
  - Click a predefined gradient.
  - Click **More Gradients** to open the style selection dialog box and configure a gradient.
  - Click **Patterns** to open the style selection dialog box and select a pattern.
  - Click **Textures** to open the style selection dialog box and select a texture.

For more information about the style selection dialog box, see *Setting Style* on page 103.

## Setting the Text Alignment

You can change the horizontal and vertical positioning of text within a text box element or button element.

You can also change the positioning for a text element. If the text is modified at design time or run time, the alignment sets how the element boundary changes to fit around the modified text.

---

**Note:** You can also set the text alignment in the **Properties Editor** by setting the **Alignment** property.

---

If the element is a text box or a button, then the text is aligned accordingly.

If the element is a text element and you then modify the text at design time or run time, the text is anchored to the point of alignment.

- Text right alignments move additional text further over to the left.
- Text left alignments move additional text to the right.
- Changes in font size leave the point of alignment unchanged and modify the frame accordingly.

**To set the text alignment**

1. Select a text element, text box element or button element on the canvas.
2. On the **Format** menu, point to **Text Alignment**, and then click the appropriate command:

---

<b>Click this command</b>	<b>To</b>
<b>Top Left</b>	Align the text at the top left frame handle.
<b>Top Center</b>	Align the text at the top middle frame handle.
<b>Top Right</b>	Align the text at the top right frame handle.
<b>Middle Left</b>	Align the text at the middle left frame handle.
<b>Middle Center</b>	Align the text in the middle of the element.
<b>Middle Right</b>	Align the text at the middle right frame handle.
<b>Bottom Left</b>	Align the text at the bottom left frame handle.
<b>Bottom Center</b>	Align the text at the bottom center frame handle.
<b>Bottom Right</b>	Align the text at the bottom right frame handle.

---

## Substituting Strings

You can search and replace strings of any element that have the Text property on your canvas. You can use the basic mode to replace strings in a list.

You can also use advanced functions, such as find and replace, ignore, case-sensitivity, and wildcards.

You cannot substitute static strings that are used in an Radio Button Group, List Box or Combo Box.

If you substitute strings for a text element in an embedded graphic, that text element is not updated if you change the source graphic's text. For example, an embedded graphic contains a text graphic with the string "SomeTextHere". You substitute "SomeTextHere" with "MyText", and then change the source graphic text from "SomeTextHere" to "NewText". The text in the embedded graphic will still show "MyText".

### To substitute strings in a graphic by using the list

1. Select one or more elements.
2. Do one of the following:
  - Press Ctrl + L.
  - On the **Special** menu, click **Substitute Strings**.

The **Substitute Strings** dialog box appears.

3. In the **New** column, type the text to be replaced.
4. Click **OK**.

## To substitute strings in a graphic by using advanced functions

1. Select one or more elements.
2. Do one of the following:
  - Press **Ctrl + E**.
  - On the **Special** menu, click **Substitute Strings**.  
The **Substitute Strings** dialog box appears.
3. Click **Find & Replace**. The dialog box expands and shows advanced options.
4. Configure the search strings. Do any of the following:
  - To find specific strings in the list, type a string in the **Find What** box and click **Find Next** to find the next string.
  - To replace a selected found string with another string, type a string in the **Replace with** box and click **Replace**.
  - To replace multiple strings, type values in the **Find What** and **Replace with** boxes and click **Replace all**.
5. Configure the search options. Do any of the following:
  - If you want the search to be case-sensitive, click **Match Case**.
  - To find only entire words that match your search string, click **Match Whole Word Only**.
  - To use wildcards, click **Use Wildcards**. Use an asterisk (\*) to search for any sequence of characters. Use a question mark (?) to search for strings with one variable character.
6. Click **OK**.

## Setting Style

You can set the fill, line, and text style from various places in the Industrial Graphic Editor using the style selection dialog box. The style selection dialog box is common to any element for which you can set a solid color, gradient, pattern, or texture. You can also set the transparency of the style.

Because you can open the style selection dialog box from different places in the Industrial Graphic Editor, the dialog box header can be different.

Also, not all tabs may be available. For example, for setting one color of a gradient in the Properties Editor, you can only select a solid color from the style selection dialog box.

---

**Note:** For information about working with legacy graphics, see *Loading Graphics with Deprecated Features* on page 112.

---

## Setting a Solid Color

You can set a solid color using the **Solid Color** tab in the style selection dialog box. You can set a solid color from the:

- Standard palette.
- Color disc and bar.
- Value input boxes.
- Color picker.
- Custom palette.

You can also:

- Add the new color to the custom palette.
- Remove a color from the custom palette.
- Save the custom palette.
- Load a custom palette.

## Setting a Solid Color from the Standard Palette

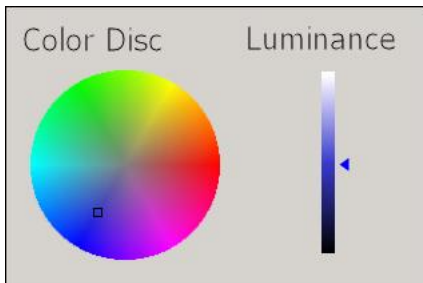
You can set a solid color from the standard palette using the **Solid Color** tab in the style selection dialog box. The standard palette is a set of 48 predefined colors you can use to quickly select a solid color.

### To set a solid color from the Standard Palette

1. In the style selection dialog box, click the **Solid Color** tab.
2. In the **Standard Palette** area, click a color. The new color appears in the **New** color box on the right of the dialog box.
3. Click **OK**.

## Setting a Solid Color from the Color Disc and Bar

You can set a solid color using the color disc and bar on the **Solid Color** tab in the style selection dialog box. The color disc and bar let you graphically select the color and the luminance (brightness).



### To set a solid color from the color disc and bar

1. In the style selection dialog box, click the **Solid Color** tab.
2. Click on the color disk to select a color. The bar is updated and shows the selected color in varying degrees of luminance (brightness).
3. Click on the bar to select a luminance (brightness). The new color appears in the **New** color box on the right of the dialog box.
4. Click **OK**.

## Setting a Solid Color with the Value Input Boxes

You can set a solid color by typing values that define the color, such as:

- Red component (0-255).
- Green component (0-255).
- Blue component (0-255).
- Hue (0-255).
- Saturation (0-255).
- Luminance (0-255).



### To set a solid color with the value input boxes

1. In the style selection dialog box, click the **Solid Color** tab.
2. In the **Red**, **Green**, **Blue**, **Hue**, **Sat.** and **Lum.** boxes, type respective values. The resulting color appears in the **New** color box on the right of the dialog box and also on the color wheel and bar.
3. Click **OK**.

## Setting a Solid Color with the Color Picker

You can set a solid color by using the color picker on the **Solid Color** tab in the style selection dialog box. The color picker lets you select a color from anywhere on the screen, even outside the IDE application.

### To set a solid color with the color picker

1. In the style selection dialog box, click the **Solid Color** tab.
2. Click the **Color Picker** button. The color picker pointer appears.
3. Select a color from anywhere on the screen by moving the mouse. As you move the mouse, the new color appears in the **New** color box on the right of the dialog box.
4. Click the mouse to complete the color selection.
5. Click **OK**.

## Setting a Solid Color from the Custom Palette

You can set a solid color from the custom palette on the **Solid Color** tab in the style selection dialog box. The custom palette is a set of colors that you want to frequently use. You can save the custom palette to a .pal file or load a custom palette from a .pal file.

To use colors from the custom palette, you must first add them. For more information, see *Adding and Removing Colors in the Custom Palette* on page 105.

### To set a solid color from the custom palette

1. In the style selection dialog box, click the **Solid Color** tab.
2. In the **Custom Palette** area, select a color. The new color appears in the **New** color box on the right of the dialog box.
3. Click **OK**.

## Adding and Removing Colors in the Custom Palette

You can add up to 36 solid colors to the custom palette. You can also remove any colors from the custom palette.

You cannot add a color that is already in the custom palette.

### To add a solid color to the custom palette

1. In the style selection dialog box, click the **Solid Color** tab.
2. Add the color. Do any of the following:
  - Select a solid color from the custom palette.
  - Select a solid color from the color disc and bar.
  - Type values for red, green, blue, hue, saturation, and luminance.
  - Select a solid color with the color picker.

The new solid color appears in the **New** color box on the right of the dialog box.

3. Click the add button above **Custom Palette**. The solid color is added to the **Custom Palette** area.

#### To remove a solid color from the custom palette

1. In the style selection dialog box, click the **Solid Color** tab.
2. In the **Custom Palette** area, select the solid color you want to remove.
3. Click the delete button above **Custom Palette**. The solid color is removed from the custom palette.

## Saving and Loading the Custom Palette

You can save the current custom palette or load a previously saved custom palette. The custom palette is loaded from or saved to a Windows Palette file (.pal).

After you save or load a custom palette, the .pal file is not connected to the graphic in any way.

#### To save a custom palette

1. In the style selection dialog box, click the **Solid Color**.
2. Click the **Save Palette** button. The **Save Palette** dialog box appears.
3. Browse to the location where you want to save the custom palette, type a name, and then click **Save**. The custom palette is saved as a palette file.

#### To load a custom palette

1. In the style selection dialog box, click the **Solid Color** tab.
2. Click the **Load Palette** button.
3. If you currently have colors in the custom palette, a message appears. Click **Yes** to continue and overwrite the current colors in the custom palette.
4. In the **Load Palette** dialog box, browse to the location of the palette file, select it, and then click **Open**. The custom palette is loaded from the selected file.

## Setting a Gradient

You can configure gradients by the:

- Number of colors - 1, 2 or 3.
- Direction - horizontal, vertical, radial, or customized.
- Variant - depending on your selection for the number of colors and direction.
- Color distribution shape - triangular with options to configure the center and falloff.
- Focus scales - width and height.

You set a gradient on the **Gradient** tab in the style selection dialog box.

## Setting the Number of Colors for a Gradient

You can set the number of colors you want to use in a gradient.

- If you use one color, the gradient is between this solid color and a specified shade of black to white.
- If you use two colors, the gradient is between these two colors.
- If you use three colors, the gradient is between these three colors in sequence.

#### To set a gradient using one color

1. In the style selection dialog box, click the **Gradient** tab.

2. In the **Colors** area, click **One**. A color selection box and a slider for the dark to light selection appears.
3. Click the color selection box to open the **Select Solid Color 1** dialog box. Select a solid color and click **OK**. For more information about this dialog box, see *Setting a Solid Color* on page 103.
4. Move the slider between **Dark** and **Light**. The new gradient appears in the **New** color box on the right of the dialog box.
5. Click **OK**.

#### To set a gradient using two colors

1. In the style selection dialog box, click the **Gradient** tab.
2. In the **Colors** area, click **Two**. Two color selection boxes appear.
3. Click the **Color 1** or **Color 2** color field to select a color from the style selection dialog box. For more information about this dialog box, see *Setting a Solid Color* on page 103.  
The new gradient appears in the **New** color box on the right of the dialog box.
4. Click **OK**.

#### To set a gradient for three colors

1. In the style selection dialog box, click the **Gradient** tab.
2. In the **Colors** area, select **Three**. Three color selection boxes appear.
3. Click the **Color 1**, **Color 2** or **Color 3** color field to select a color from the style selection dialog box. For more information about this dialog box, see *Setting a Solid Color* on page 103.  
The new gradient appears in the **New** color box on the right of the dialog box.
4. Click **OK**.

## Setting the Direction of the Gradient

You can configure the direction of the gradient to be one of the following:

- Horizontal - from side to side
- Vertical - up and down
- Radial - circular from the center outwards
- Custom angle - across the element at a specified angle

#### To set a horizontal gradient

1. In the style selection dialog box, click the **Gradient** tab.
2. In the **Direction** area, click **Horizontal**. The new gradient appears in the **New** color box on the right of the dialog box.
3. Click **OK**.

#### To set a vertical gradient

1. In the style selection dialog box, click the **Gradient** tab.
2. In the **Direction** area, click **Vertical**. The new gradient appears in the **New** color box on the right of the dialog box.
3. Click **OK**.

#### To set a radial gradient

1. In the style selection dialog box, click the **Gradient** tab.

2. In the **Direction** area, click **Radial**.
3. Set the center location. Do any of the following:
  - In the **Horizontal** and **Vertical** boxes, type values for the center location.
  - Click and drag the center point in the adjacent box.The new gradient appears in the **New** color box on the right of the dialog box.
4. Click **OK**.

#### To set the custom angle of a gradient

1. In the style selection dialog box, click the **Gradient** tab.
2. In the **Direction** area, click **Custom**.
3. Set the angle. Do any of the following:
  - In the **Angle** text box, type a value for the angle.
  - Click and drag the angle bar in the adjacent box.The new gradient appears in the **New** color box on the right of the dialog box.
4. Click **OK**.

## Changing the Variant of a Gradient

You can change the variant of a gradient. The variants are alternate gradients with the same colors you can quickly select.

#### To change the variant of a gradient

1. In the style selection dialog box, click the **Gradient** tab.
2. In the **Variants** area, click on a variant gradient.  
The new gradient appears in the **New** color box on the right of the dialog box.
3. Click **OK**.

## Setting the Color Distribution Shape

You can configure the distribution shape of a triangle gradient with one or two colors.

- In a triangular distribution, the gradient from one color to the next rises and falls at the same rate.

You can also configure the peak and the falloff.

- The peak specifies the offset of the gradient if it has one or two colors.
- The falloff specifies the amplitude of the gradient if it has one or two colors.

Additionally, you can configure the center point of a radial gradient if it is defined by three colors.

#### To use a triangular gradient

1. In the style selection dialog box, click the **Gradient** tab.
2. In the **Color Distribution Shape** area, click **Triangular**. The new gradient appears in the **New** color box on the right of the dialog box.
3. Click **OK**.

#### To set the peak of a gradient with one or two colors

1. In the style selection dialog box, click the **Gradient** tab.
2. In the **Color Distribution Shape** area, do one of the following:

- Use the **Peak** slider to specify the peak.
- In the **Peak** box, type a value from 0 to 100.

The new gradient appears in the **New** color box on the right of the dialog box.

3. Click **OK**.

### To set the falloff of a gradient with one or two colors

1. In the style selection dialog box, click the **Gradient** tab.
2. In the **Color Distribution Shape** area, do one of the following:
  - Use the **Falloff** slider to specify the peak.
  - In the **Falloff** box, type a value from 0 to 100.

The new gradient appears in the **New** color box on the right of the dialog box.

3. Click **OK**.

### To set the center point of a radial gradient with three colors

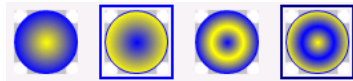
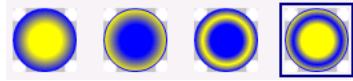
1. In the style selection dialog box, click the **Gradient** tab.
2. In the **Color Distribution Shape** area, do one of the following:
  - Use the **Center** slider to specify the peak.
  - In the **Center** box, type a value from 0 to 100.

The new gradient appears in the **New** color box on the right of the dialog box.

3. Click **OK**.

## Setting the Focus Scales of a Gradient

You can set the focus scales of a radial gradient. The focus scales acts as a magnification of the gradient. You can set the height and width of the focus scales.

Height	Width	Appearance
0	0	
50	50	

### To set the height and width of the focus scales for a gradient

1. In the style selection dialog box, click the **Gradient** tab.
2. In the **Focus Scales** area, do one of the following:
  - Move the **Height & Width** slider to specify the height and width.
  - In the text box, type the value for the height and width.

The new gradient appears in the **New** color box on the right of the dialog box.

3. Click **OK**.

## Setting a Pattern

You can set a pattern for an element. The following table describes the pattern options:

Pattern	Options
Horizontal	Simple, Light, Narrow, Dark, Dashed
Vertical	Simple, Light, Narrow, Dark, Dashed
Percent	05, 10, 20, 25, 30, 40, 50, 60, 70, 75, 80, 90
Grid	Small, Large, Dotted
Checker Board	Small, Large
Diagonals	Forward, Backward, Dashed Upward/Downward, Light/Dark/Wide Upward/Downward
Diamond	Dotted, Outlined, Solid
Cross	Diagonal
Brick	Horizontal, Diagonal
Confetti	Small, Large
Others	Zig Zag, Wave, Weave, Plaid, Divot, Shingle, Trellis, and Sphere

Patterns consist of the foreground color and the background color that you can change.

### To set a pattern

1. In the style selection dialog box, click the **Pattern** tab.
2. Select a pattern. The new pattern appears in the **New** color box on the right of the dialog box.
3. If you want to change the foreground color of the pattern, click the **Foreground** color selection box. The style selection dialog box appears. Select a solid color and click **OK**.
4. If you want to change the background color of the pattern, click the **Background** color selection box. The style selection dialog box appears. Select a solid color and click **OK**.

For more information about setting a solid color, see *Setting a Solid Color* on page 103.

5. Click **OK**.

## Setting a Texture

Textures are images you can use as styles for lines, fills and text. You can stretch the image or tile the image across the entire element to be filled.

### To set a texture

1. In the style selection dialog box, click the **Textures** tab.
2. Click **Select Image**. The **Open** dialog box appears. You can import the following image formats: .BMP, .GIF, .JPG, .JPEG, .TIF, .TIFF, .PNG, .ICO, .EMF. Animated GIF images are not supported.
3. Browse to and select an image file and click **Open**. The new pattern appears in the **New** color box on the right of the dialog box.
4. Configure the size mode. Do one of the following:

- Click **Tile** to create a pattern that repeats itself.
  - Click **Stretch** to enlarge (or shrink) the pattern across the selected element.
5. Click **OK**.

## Setting the Style to No Fill

You can set the style to "No Fill". For example if you set the fill style of a rectangle element to No Fill, the background of the rectangle appears transparent.

### To set the No Fill style

1. In the style selection dialog box, click the **No Fill** tab.  
The No Fill style appears as a red cross-through line in the **New** color box on the right of the dialog box.
2. Click **OK**.

## Setting the Transparency of a Style

You can set the transparency of a solid color, gradient, pattern, or texture.

### To set the transparency of a style

1. Open the style selection dialog box.



2. At the bottom of the dialog box, do one of the following:
  - Drag the **Transparency** slider handle left or right to change the transparency percentage.
  - In the **Transparency** text box, type a percentage value.The new style appears in the **New** color box.
3. Click **OK**.

## Setting the Transparency Level of an Element

You can set the transparency level of an element. Levels range from 0 percent (opaque) to 100 percent (transparent).

Transparency of a group of elements behaves in a special way.

### To set the transparency level of an element

1. Select one or more elements.
2. On the **Format** menu, click **Transparency**.
3. To use a predefined level, select it from the list.
4. To use a different level, click **More Transparency Levels**. The **Select Transparency Level** dialog box appears. Type a transparency level in the **Transparency** text box or use the slider to select a transparency level.
5. Click **OK**.

---

**Note:** You can also set the transparency level by changing the **Transparency** property in the Properties Editor.

---

## Adjusting the Colors and Transparency of a Gradient

You can easily change the colors and transparency of an element with a gradient style.

For example, you can create pipes with a gradient style of different colors. You can change the pipe color, but still keep the 3-D appearance.

You do this in the Properties Editor using the Color1, Color2, Color3, and Transparency sub-properties.

### To tweak the colors and transparency of a gradient

1. Select the element for which you want to change colors or transparency.
2. In the Properties Editor, locate the appropriate style setting. This can be:
  - FillColor
  - LineColor
  - TextColor
  - UnFillColor
3. Click the + icon to expand the property. The Color1, Color2, Color3, and Transparency sub-properties are shown.
4. Do one of the following:
  - Click the color box of one of the color sub-properties.
  - Type a new value for the transparency and click **Enter**.
5. Click the browse button. The style selection dialog box appears.
6. Select a color from the style selection dialog box and click **OK**. The solid color is applied to the selected element.

## Loading Graphics with Deprecated Features

In recent versions of Microsoft's rendering technologies, certain gradient features have been deprecated. To accommodate and future proof graphics built using the Industrial Graphic Editor, the affected features have been removed from the configuration environment. Graphics previously configured with deprecated features will continue to render as expected.

Deprecated gradients are as follows:

- Point Based gradient direction
- Bell gradient shape (1 or 2 color selection)
- Radial gradient direction without locked focus

When working with graphics that have been configured with deprecated gradients, the Gradient tab in the style selection dialog box shows Point Based direction, Bell color distribution, and the Focus Scales lock option disabled. You can choose an available option to enable other available options, and save your configuration to update the graphic.

## Enabling and Disabling Elements for Run-Time Interaction

You can enable or disable elements so that the run time user cannot use any interaction animations, such as:

- User input.
- Horizontal and vertical sliders.
- Pushbuttons.



- Action scripts.
- Showing and hiding graphics.

Other animations such as horizontal fills and tooltips continue to work as expected.

#### **To enable an element for run-time interaction**

1. Select one or more elements you want to enable.
2. In the Properties Editor **Runtime Behavior** group, set the Enabled property to True.

#### **To disable an element for run-time interaction**

1. Select one or more elements you want to disable.
2. In the Properties Editor **Runtime Behavior** group, set the Enabled property to False.

## Changing the Visibility of Elements

You can configure elements to be hidden or shown at run time.

The visibility of an element does not affect its animations. Even when an element is invisible, its animations continue to be evaluated.

#### **To configure an element to be shown at run time**

1. Select one or more elements you want to have shown at run time.
2. In the Properties Editor **Runtime Behavior** group, set the Visible property to True.

#### **To configure an element to be hidden at run time**

1. Select one or more elements you want to have hidden at run time.
2. In the Properties Editor **Runtime Behavior** group, set the Visible property to False.

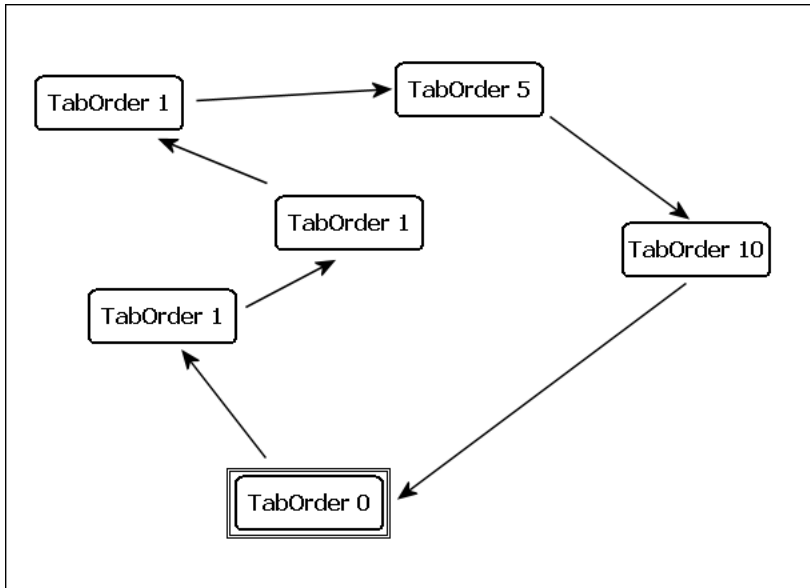
## Editing the Tab Order of an Element

You can configure the elements on the canvas so that at run time you can use the Tab key to put each element in focus in a specified sequence. This sequence is called the tab order.

By default, when you place elements on the canvas, they have a tab order number of 0. Elements with the same tab order number are placed into focus by tabbing at run time according to their z-order. This means they are tabbed through at run time according to their position in the Elements List.

You can override the tab order by assigning a unique index number to the TabOrder property of each element.

Lower tab order numbers take precedence over higher tab order numbers. You must change this value to determine the tab order sequence.



You must also make sure that the **TabStop** property of each element is set to true. When the **TabStop** property is set to true, you can use the **Tab** key at run time to switch to the selected element.

#### To edit the element's tab order

1. Select the element for which you want to set the tab order.
2. In the Properties Editor, ensure that the **TabStop** property is set to True.
3. Type a unique value for the **TabOrder** property.

## Using the Format Painter to Format Elements

You can apply formatting of one element to other elements quickly by using the format painter. You can apply the format of one element:

- One time to other elements.
- In repetitive mode to other elements.

When you use the format painter, it copies the following formats of the element if applicable to the target elements:

- Font family, size, and style
- Text style, alignment, and word wrap settings
- Line style, weight, pattern, and ends
- Transparency
- Fill style, orientation, behavior, horizontal percent fill, and vertical percent fill
- Unfilled style
- Horizontal and vertical direction properties

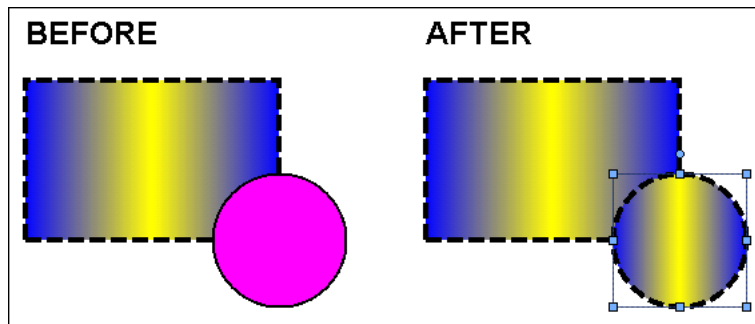
You cannot use the format painter for:

- The status element
- An element that is part of a path

- Groups of elements
- Elements in different hierarchy groups

### To copy the format of an element one time

1. Select the element with the format you want to copy.
2. On the **Edit** menu, click **Format Painter**. The pointer appears as the format painter cursor.
3. Select the element you want to apply the format to. The format is applied to the clicked element.



### To copy the format of an element in repetitive mode

1. Select the element with the format you want to copy.
2. On the toolbar, double-click the Format Painter icon. The pointer appears as the format painter cursor.
3. Click each element you want to apply the format to. The format is applied to the clicked element.
4. Repeat Step 3 for any other elements you want to apply the format to.
5. When you are done, press the Esc key.

## Editing the General Properties of a Graphic

You can configure the general properties of a graphic. The general properties determine the overall appearance and behavior of the graphic. You can:

- Add a meaningful description to your graphic.
- Enable anti-aliasing, or smoothing, for your graphic to improve its appearance. The anti-aliasing filter essentially blurs the elements slightly at the edges.
- Allow or prevent the opening of more than one graphic or display from a graphic. One example is a graphic with multiple Show Symbol animations. If this option is enabled, you can open more than one pop-up and each pop-up is modeless.

### To edit the description of a graphic

1. Click on the canvas so that no elements are selected.
2. In the Properties Editor, type a meaningful description for the **Description** property.

### To use smoothing (anti-aliasing) for a graphic

1. Click on the canvas so that no elements are selected.
2. In the Properties Editor, select True for the **Smoothing** property.

### To enable multiple pop-ups for a graphic

1. Click on the canvas so that no elements are selected.
2. In the Properties Editor, select True for the **MultiplePopupsAllowed** property.



# CHAPTER 6

## Editing Graphic-Specific and Element-Specific Properties

### About Graphic- and Element-Specific Properties

You can configure graphic-specific and element-specific properties. For properties that are common to all or most elements, see *Editing Common Properties of Elements and Graphics* on page 95.

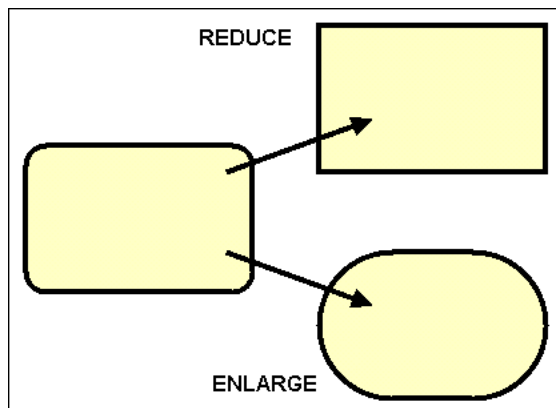
You can configure:

- General properties of a graphic.
- Radius of rounded rectangles.
- Shape and end appearance of lines and H/V lines.
- Auto-sizing and word-wrapping in text boxes.
- Image-specific properties.
- Button-specific properties.
- Control points and tension in curves.
- Angles in pies, chords, and arcs.
- Status elements.
- Windows common controls.

### Setting the Radius of Rounded Rectangles

You can specify the radius, in pixels, of the corners of rounded rectangles. The radius determines their "roundness". You can:

- Enlarge or reduce the radius of the rounded rectangle on the fly. The easiest way to do this is with the keyboard.
- Set the radius of the rounded rectangle to a specific value using the Properties Editor.



Rounded rectangles maintain their radius when their size is changed. If the graphic containing rounded rectangles is embedded into a window and resized, the radius is not affected. This can have adverse effects on the graphic representation of your graphic.

### To enlarge the radius of a rounded rectangle

1. Select one or more rounded rectangles on the canvas.
2. Press and hold Shift and the + key on the number pad. The radius is enlarged, and the rounded rectangle becomes more round.

### To reduce the radius of a rounded rectangle

1. Select one or more rounded rectangles on the canvas.
2. Press and hold Shift and the minus (-) key on the number pad. The radius is reduced, and the rounded rectangle becomes more rectangular.

### To set the radius of a rounded rectangle exactly

1. Select one or more rounded rectangles on the canvas.
2. In the Properties Editor, change the value for Radius property and click **Enter**. The selected rounded rectangles are updated accordingly.

## Setting Line End Shape and Size

You can set the line end shape and size for any element that contains open lines such as lines, H/V lines, polylines, curves, and arcs.

For a line end, you can set the shape to be an arrowhead, diamond, circle, or square. You can set the size if the line end shape is an arrowhead.

### To set the line end shape

1. Select one or more elements.
2. On the **Format** menu, click **Line Ends**.
3. To use a predefined line end shape, select it from the list.
4. To use another line shape, click **More Line Options**. The **Select Line Options** dialog box appears.
5. Do the following:
  - a. In the **Line Start** list, click a shape for the start of the line.
  - b. In the **Line End** list, click a shape for the end of the line.
  - c. Click **OK**.

### To set the size of the line arrowheads

1. Select one or more open line elements.
2. On the **Format** menu, click **More Line Options**. The **Select Line Options** dialog box appears.
3. Select a size on the **Line Start Size** list if the line starts with an arrowhead. Valid sizes are: XX Small, X Small, Small, Medium Small, Medium, Medium Large, Large, X Large, XX Large.
4. Select a size on the **Line End Size** list if the line ends with a shape.
5. Click **OK**.

---

**Note:** You can also set the line end shapes by changing the **StartCap** and **EndCap** properties in the Properties Editor.

---

## Setting Auto Scaling and Word Wrapping for a Text Box

You can configure a text box to auto scale the text or to word wrap the text within the text box.

- For auto scaling, the text is resized to fit the text box.
- For word wrapping, the text in a text box continues on the next line.

### To auto scale the text in a text box

1. Select one or more text boxes.
2. In the Properties Editor, set the AutoScale property to true.

### To word wrap the text in a text box

1. Select one or more text boxes.
2. In the Properties Editor, set the WordWrap property to true.

## Using Images

You can place images on the canvas. This is a two step process:

1. Draw a frame which specifies the target size of the image.
2. Import the image from an image file.

After you place an image on the canvas, you can:

- Set the display mode (ImageStyle).
- Set the image alignment (ImageAlignment).
- Set the transparency color (HasTransparentColor, TransparentColor properties).
- Open the image in an image editing application.
- Select a different image for the image element.

## Placing an Image on the Canvas

You can place an image on the canvas. The image data must come from an image file. You can import the following image formats: .BMP, .GIF, .JPG, .JPEG, .TIF, .TIFF, .PNG, .ICO, .EMF.

You cannot use animated GIF images.

### To place an image on the canvas

1. In the **Tools** panel, select the image icon.
2. Click the canvas where you want to place the image and drag the mouse to draw a rectangle that will contain your image.
3. Release the mouse button. The **Open** dialog box appears.
4. Browse to and select an image file, and then click **Open**. The image is loaded into the image frame.

If the image frame is smaller than the image, the image is cropped to fit into the frame. If the image frame is larger than the image, the image appears in its original size.

## Setting the Image Display Mode

You can set the way the image appears on the canvas.

- In normal mode, the image is not stretched or tiled. You can resize the image frame with the resizing handles.

- In stretch mode, the image is stretched so that it fills its frame.
- In tile mode, the image is repeated so that a tiled pattern that fills its frame is created.
- In auto mode, the image frame is enlarged or reduced to the image size. The resizing handles are locked. When the image style of an image element is Auto, you cannot change its size.

#### To stretch an image to the image frame

1. Select the image element you want to stretch.
2. In the Properties Editor, select **ImageStyle**.
3. In the list, click **Stretch**. The image is stretched to the image frame.

#### To tile an image in an image frame

1. Select the image element you want to tile.
2. In the Properties Editor, select **ImageStyle**.
3. In the list, click **Tile**. The image is tiled to fill the image frame.

#### To set an image frame size to its image size

1. Select the image element you want to adjust.
2. In the Properties Editor, select **ImageStyle**.
3. In the list, click **Auto**. The image frame is enlarged or reduced to the image size.

## Setting the Image Alignment

The image alignment specifies where the image appears in an image frame. By default, images appear in the center of the image frame. You can change this setting to one of the following:

- Top left, top center, or top right
- Middle left, center, or middle right
- Bottom left, bottom center, or bottom right

---

**Note:** You can also set the image alignment in the **ImageAlignment** property in the Properties Editor.

---

#### To set the image alignment

1. Select the image element with the image you want to align.
2. In the Properties Editor, select **ImageAlignment**.
3. In the list, click one of the following options: TopLeft, TopCenter, TopRight, MiddleLeft, Centers, MiddleRight, BottomLeft, BottomCenter or BottomRight. The image is aligned accordingly in the image frame.

## Setting the Image Color Transparency

You can use image color transparency to specify that a color within an image is partially or entirely transparent. When you configure image transparency, you must:

- Enable color transparency for images.
- Specify the color that is to be transparent.

Setting the image color transparency is different than setting the transparency of the image element, as it only applies to one color. Image transparency applies to the entire image.



**To enable image color transparency**

1. Select the image element.
2. In the Properties Editor, select **HasTransparentColor**.
3. In the list, click **True**.

**To set the transparency color for an image**

1. Select the image element.
2. On the **Edit** menu, click **Select Image Transparent Color**. The pointer becomes a color picker.
3. Click the color you want to use as the transparency color. The image is updated with the new transparency color.

---

**Note:** You can also select a transparency color with the **TransparentColor** property in the Properties Editor. For more information about setting the color, see *Setting a Solid Color* on page 103.

---

## Editing the Image

You can edit the image in an image element by opening it in an image editing application.

You can specify the image editor by changing the designer preferences. For more information, see *Setting the Image Editing Application* on page 121.

**To edit an image**

1. Select the image element with the image you want to edit.
2. On the **Edit** menu, click **Edit Image**. The image is opened with the associated image editing application.
3. Make changes to the image as needed, save the image and close the image editing application. The image is updated on the canvas.

## Setting the Image Editing Application

You can specify the image editor that opens when you select an image for editing. You can select a currently registered image editing application or add one.

**To set the image editing application**

1. On the **Special** menu, click **Preferences**. The **Designer Preferences** dialog box appears.
2. Select an image editor from the Image Editor list.

**To add an image editing application**

1. On the **Special** menu, click **Preferences**. The **Designer Preferences** dialog box appears.
2. In the **Image Editor** list, click **Choose Custom Editor**. The **Select Image Editing Application** dialog box appears.
3. Browse to and select the executable file of the image editing application and click **Open**. The image editor is added to the list.

## Selecting a Different Image

You can change the current image of an image element by selecting a new image.

**To select a different image**

1. Select the image element with the image you want to change.

2. On the **Edit** menu, click **Select Image**. The **Open** dialog box appears.
3. Browse to and select an image file, and then click **Open**. The image is loaded into the image frame.

---

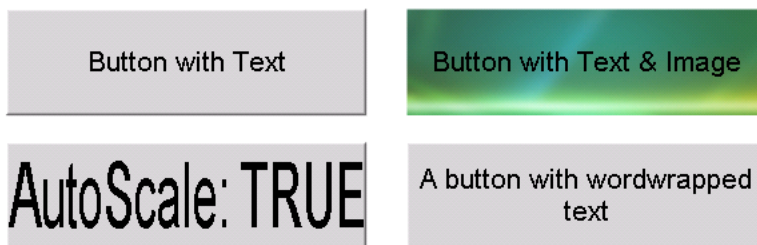
**Note:** You can also select a different image by clicking the browse button in the **Image** property in the Properties Editor .

---

## Using Buttons

You can add a text caption or an image to buttons that belong to Industrial Graphics. If a button includes a text caption, you can:

- Automatically scale the font size
- Configure the text to wrap within the button



## Automatically Scaling Text in Buttons

You can automatically scale text so that the font size is adapted to the button size.

### To automatically scale text in buttons

1. Select the button element on the canvas.
2. In the Properties Editor, set the **AutoScale** property to True.

## Wrapping Text in Buttons

You can wrap text in buttons.

### To wrap text in buttons

1. Select the button element on the canvas.
2. In the Properties Editor, set the **WordWrap** property to True.

## Configuring Buttons with Images

You can use buttons with an image in Industrial Graphics.

- The "up" image appears after a button is released and returns to the up position during run time
- The "down" image appears after a button is pressed and locks in the down position during run time

You can edit an up image or a down image after you assign it to a button.

### To use a down image or up image on a button

1. Select the button element on the canvas.
2. In the Properties Editor, select **Image** in the property **ButtonStyle** list.
3. Click the browse button of the **UpImage** property and select an image in the **Open** dialog box. This is the image that appears on the button by default and also when the button is released.

4. Click the browse button of the **DownImage** property and select an image in the **Open** dialog box. This image appears after the button is clicked.

### To edit an up image or a down image of a button

1. Right-click the button element on the canvas. The context menu appears.
2. Click **Edit Button Image**, then click one of the following:
  - Edit Up Image
  - Edit Down Image

The up image or down image is opened in the default image editor.

3. Edit the image.
4. Save the image and close the image editor. The up image or down image is updated.

## Editing Control Points

Control points determine the shapes of polylines, polygons, curves, and closed curves. To change the shape of these elements after they have been placed on the canvas, you can:

- Move individual control points.
- Add or remove control points.

## Moving Control Points

After you place a polyline, polygon, curve, or closed curve on the canvas, you can change its shape by editing its control points.

### To move the control points of a polyline, polygon, curve, or closed curve

1. Select the polyline, polygon, curve, or closed curve.
2. On the **Edit** menu, click **Edit Control Points**. The control points of the element are shown.
3. Click a control point you want to change and drag it to the new location. The element is updated accordingly.
4. Repeat the previous step for all control points you want to change.

## Adding and Removing Control Points

You can add or remove control points from polylines, polygons, curves, and closed curves.

### To add control points to a curve or closed curve

1. Select the curve or closed curve.
2. On the **Edit** menu, click **Edit Control Points**. The control points of the element are shown.
3. Press and hold the Shift key.
4. Move the mouse over the curve or closed curve at the point you want to add a control point. The pointer appears as a pen with a plus graphic.
5. Click the curve or closed curve. The control point is added to the curve or closed curve.
6. Repeat the last step for any other control points you want to add.
7. When you are done, release the Shift key.

### To delete control points from a curve or closed curve

1. Select the curve or closed curve.

2. On the **Edit** menu, click **Edit Control Points**. The control points of the element are shown.
3. Press and hold the Shift key.
4. Move the mouse over the control point you want to remove. The pointer appears as a pen with a minus graphic.
5. Click the control point. The control point is removed from the curve or closed curve.
6. Repeat the last step for any other control points you want to remove. You must have at least two control points.
7. When you are done, release the Ctrl key.

## Changing the Tension of Curves and Closed Curves

After you place a curve or a closed curve, you can change its tension. The tension specifies how tightly the curve bends through the control points. Valid range are float values from 0 (tightly) to 2 (loosely).

---

**Note:** You can also change the tension of a curve or closed curve by changing the value for the Tension property in the Properties Editor.

---

### To edit the tension of a curve or closed curve

1. Select the curve or closed curve.
2. In the Properties Editor, type a float value from 0 to 2 for the **Tension** property.

## Changing Angles of Arcs, Pies and Chords

After you place an arc, pie, or chord, you can change the start and sweep angles of elements. You can change the angles to any integer degree from 0 to 359. When you change the angles, you can press the Shift and Ctrl keys to make the angle snap to multiples of 15 or 45 degrees.

You can also move the start angle and sweep angle at the same time. The object appears to be rotated around its arc/pie/chord center point while keeping the same center point angle.

---

**Note:** You can also change the start or sweep angle of an arc, pie or chord in the **StartAngle** or **SweepAngle** properties in the Properties Editor. For more information, see *Utilizing Sweep Angle Run-Time Properties* on page 125.

---

### To change the start or sweep angle of an arc, pie, or chord

1. Select the arc, pie, or chord.
2. On the **Edit** menu, click **Edit Start and Sweep Angles**. The start and sweep angle handles appear on the selected element.
3. If you want to the angle to be multiples of 15 degrees, press and hold the SHIFT key.
4. If you want to the angle to be multiples of 45 degrees, press and hold the CTRL key.
5. Grab the start angle or the sweep angle handle and drag it to the new location. The element is updated accordingly.

### To change the start and sweep angles of an arc, pie, or chord together

1. Select the arc, pie, or chord.
2. On the **Edit** menu, click **Edit Start and Sweep Angles**. The start and sweep angle handles appear on the selected element.
3. Select the start angle or the sweep angle handle and keep the mouse button down.
4. Press and hold the Alt key.
5. If you want additionally either angles to be multiples of 15 degrees, press and hold the Shift key.

6. If you want additionally either angles to be multiples of 45 degrees, press and hold the Ctrl key.
7. Drag the mouse. The start angle and sweep angle are changed accordingly.
8. When you are done, release the mouse button and then any keys.

## Utilizing Sweep Angle Run-Time Properties

The 2 and 3 point arc, pie, and chord graphic elements contain **StartAngle** and **SweepAngle Appearance** properties. These properties can be assigned values in a client script that change during run time to show moving sweep angle or start angle lines as part of arc, pie, and chord graphic elements.

Sweep angle run-time properties are well suited for showing a current value within a range of possible values. For example, the movement of a chord sweep angle can show a pie chart with fill that indicates the current time within a repetitive period. Or, the movement of an arc sweep angle can represent a pointer to the current value within a range of possible values like a tachometer.

### To configure sweep angle run-time properties

1. Place an arc, pie, or chord graphic object on the Industrial Graphic Editor canvas.
2. Select the graphic element to show its **Properties** attributes.
3. Assign **StartAngle** and **SweepAngle** properties as values of a client script that change based on run-time events.

## Monitoring and Showing Quality and Status

You can configure your graphic to show non-good status and quality of attributes or tags in different ways:

- A status element shows a specific icon depending on the quality and status of configured attributes, tags or elements.
- The text, fill, or line appearance of elements is overridden depending on the quality and status of the attributes and tags they reference.
- Elements are drawn with an outline depending on the quality and status of the attributes they reference.

---

**Note:** Quality and status elements might not be supported by all HMIs. Refer to "Working with the Industrial Graphic Editor" in your HMI help for more information.

---

## Using Status Elements

Status elements show a specified graphic depending on the quality and status of:

- Attributes and tags configured for specific animated elements.
- One or more specified attributes or tags.

You can assign status elements to an animation in three steps:

1. Draw the status element on the canvas.
2. Associate the status element with animated elements on the canvas and/or attributes that provide the quality and status data to be monitored.
3. If needed, configure the appearance of the status element.

## Setting Number Formats by Regional Locales

The format of numbers varies by country. In the United States, a period represents the decimal point of an analog number and a comma is the thousand separator. In other countries, the purpose of the characters may be different. Germany uses a comma to represent the decimal point and a period to represent the thousand separator.

Industrial graphic numeric values can display thousands and decimal separators that match the numeric format of the country specified as the Home location of the computer running a ViewApp.

Numeric formatting by regional locale applies only to Industrial graphic number displays and input numbers included as part of your applications.

Numeric formatting by regional locale can be applied to an application containing:

- User Input animation
- Value Display animation
- Tooltip animation
- Windows Client Controls (RadioButton, ComboBox, and ListBox)
- SecuredWrite dialog
- VerifiedWrite dialog
- SignedWrite dialog

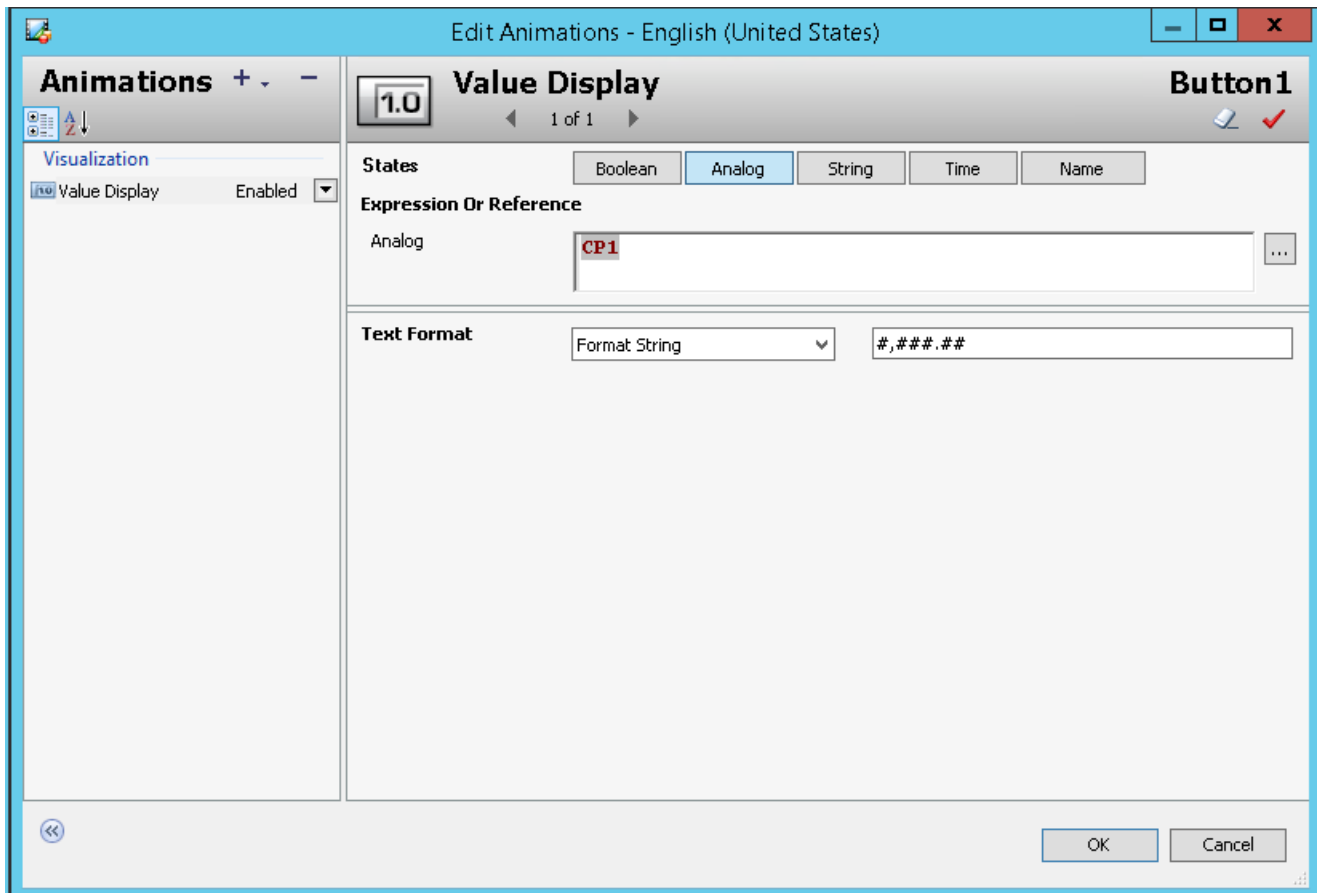
## Design Time Considerations for Numeric Formatting

During design time, you must make the following preparations to show numbers in a regional locale:

- Enter numbers according to the the U.S. format in design time, e.g. `#,###.##`
- Set the regional locale of the computer running a ViewApp

## Enter Input Numbers in U.S. Format

The Industrial Graphic Editor enables you to specify a number format during design time using a format string for Value Display animations.



A Value Display animation includes a **Text Format** field. When **Text Format** is set to **Format String**, you must enter a text string that represents the format of numbers shown during run time.

**Important:** During design time, numeric format strings must be specified in a United States format.

During design time, a numeric format string or numeric value must follow the United States number format.

- The decimal point in a number is a period.

- The thousand separator in a number is a comma.

Value in String (no format)	56000456.56
-----------------------------	-------------

Format String	User Input	Value Display
###.##	56000.456,56	56000.456,56
###,###.##	56.000.456,56	56.000.456,56
###.##	56000456,56	56000456,56
###.##	56000456,560000	56000456,560000 (wrong format)

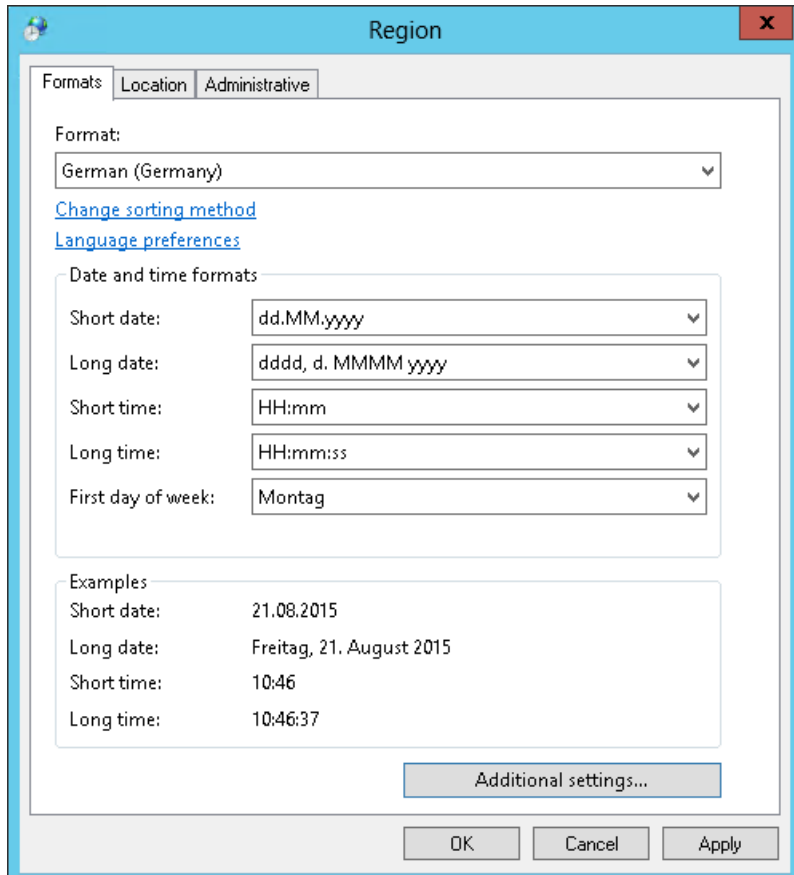
Numeric Format Styles	
	Value Display
Custom, Real	56000457
Custom, Fixed Decimal	56000456,56
Custom, Fixed Decimal, 4 precision	56000456,5600
Real	56000457
Fixed Decimal, 3 precision	56000456,56
Integer	56000457
Exponential, 3 precision	5,60e+007
Hex	4C559FF2
Binary	0100110001010101100111111110010

In the example above, the design-time numeric format strings within the box at the left comply with the U.S. number format. But during run time, User Input and Value Display animation show numbers in the correct format of the country specified by the computer's Regional setting.



## Set the Regional Locale of the Computer Hosting the HMI/SCADA Software

To enable numeric formatting by regional locale, the computer a ViewApp runs on must have its region set to the country in which you want Industrial graphic numbers to be formatted.



The **Region** setting is accessible from the Windows Control Panel. If you want to display Industrial graphic numbers in a non-U.S. format, select the **Formats** tab and select a country in the **Format** field.

## Run-Time Considerations for Formatting Numbers

The following analog number format styles support numeric formatting by regional locale during run time:

- Custom
- Real
- FixedDecimal
- Integer
- Exponential

---

**Important:** During run time, numbers are entered in the format of the selected country.

---

The following figure shows the analog number 123456.558857 formatted to the German regional locale during run time by the different numeric format styles:

<b>Numeric Format Styles</b>	<b>Value Display</b>
Custom, Real	123.457
Real	123.457
Fixed Decimal	123.456,56
Integer	123.457
Exponential	1,23e+005

A thousand separator is not required when entering a number or specifying a format string. But, if a thousand separator is used during run time, it must be placed in the correct location in a specified number or with User Input animation.

The thousand separator appears in the following numeric format styles:

- Real
- Fixed Decimal
- Integer
- Custom configured as Real, Fixed Decimal, or Integer

During run time, if your HMI/SCADA software provides software keypads, you can include comma, period, and thin space keys to enter numbers in a User Input animation data entry field that match the format of the computer's regional locale. The following illustration shows a typical software keypad. The software keypad, if supported by your HMI/SCADA software, might vary in appearance.

The screenshot displays a software keypad interface. At the top, it shows the 'Current Value' as 8.456,47. Below this, the 'Minimum Value' is 500,5 and the 'Maximum Value' is 9.000,5. A 'New Value' input field contains the text '8.456,47'. Below the input field is a numeric keypad with buttons for digits 0-9, a decimal point, a left arrow, and a right arrow. At the bottom of the keypad are 'OK' and 'Cancel' buttons.

## Restrictions of Numeric Formatting by Regional Locale

There are several restrictions of numeric formatting by regional locale that you must consider for the visualization applications you build with your HMI/SCADA software.

### Numeric Strings Enclosed Within Quotation Marks

A numeric string enclosed within quotation marks cannot be converted to the number format of another regional locale because of the ambiguity interpreting the thousand separator character.

**Example:**

"4000.654" in the U.S. regional setting is four thousand.

"4000.654" in the Germany regional setting is over four million.

---

**Important:** A numeric string is not supported and cannot be converted to the number format of another regional locale.

---

## Numbers Passed as Script Parameters

Scripts containing the SignedWrite() function experience similar problems interpreting the thousand separator character when a numeric string is passed as a parameter within quotation marks.

**Example:**

```
SignedWrite("AO1.PV1", "8.456,56", "", true, 1, null);
```

The numeric value is enclosed within quotation marks as a string data type and the comma thousand separator character is interpreted as a parameter delimiter. If the computer running a visualization application regional locale is set to Germany, the script incorrectly writes 8.46 to an attribute.

**Alternative Solution:**

Use a custom property with an analog data type instead of a string.

```
SignedWrite("AO1.PV1", CP1, "", true, 1, null);
```

where CP1=8.456,56 is set by the user at run time.

## Double-byte Character Languages

Double-byte character languages like Chinese or Japanese provide narrow or wide character sets. The Windows default setting is to show narrow characters in Chinese or Japanese languages. The decimal point and digital grouping characters can be shown with a narrow double-byte character set. However, the comma or period characters cannot be shown with a wide double-byte character set.

## Using Windows Common Controls

You can add the following Windows common controls to your graphic:

- Radio button group
- Check box
- Edit box
- Combo box
- Calendar control
- DateTime picker
- List box

You can place these Windows common controls as you would any other element by selecting them from the **Tools** panel. You click on the canvas to position a common control and, with exception of the calendar control, drag the rectangle to set of the control.

After placing the control on the canvas, you can then configure:

- Background color and text color (with exception of the DateTime Picker control).
- Other control-specific properties in the Properties Editor.
- Control-specific animations.
- The common Value property in scripting to read from and write to the Windows common control at run time.

## Changing Background Color and Text Color of Windows Common Controls

You can change the background color and text color of all Windows common controls with exception of the DateTime Picker control.

The background color and text color of the Windows common controls can be only solid colors, not gradients, patterns, nor textures.

### To set the background color of a Windows common control

1. Select the Windows common control.
2. In the Properties Editor, click the browse button of the Fill Color property. The **Select Fill Color** dialog box appears.
3. Select a solid color and click **OK**. For more information, see *Setting a Solid Color* on page 103. The Windows common control background color changes accordingly.

### To set the text color of a Windows common control

1. Select the Windows common control.
2. In the Properties Editor, click the browse button of the TextColor property. The **Select Text Color** dialog box appears.
3. Select a solid color and click **OK**. For more information, see *Setting a Solid Color* on page 103. The Windows common control text color changes accordingly.

## Reading and Writing the Selected Value at Run Time

You can use the Value property that is common to all Windows common controls. It is not visible in the Properties Editor. You can use the value property in a script or other animation links.

The following table shows you the data type, a description on how the value property is used, and an example for each Windows common control.

Control	Data Type	Description	Example
Radio Button Group	Boolean, Integer, Real or String	Reads the value of the selected item, or selects the item with that value if it exists.	<code>RadioButtonGroup1.Value = "Mixing";</code>
Check Box	Boolean	Sets or reads the checked status.	<code>CheckBox1.Value = 1;</code>
Edit Box	String	Sets or reads the text contents.	<code>EditBox1.Value = "Hello World";</code>
Combo Box	Integer	Reads the value of the selected item, or selects the item with that value if it exists.	<code>ComboBox1.Value = 5;</code>
Calendar	Time	Sets or reads the selected date.	<code>Calendar1.Value = "11/15/2006 11:12:34 AM";</code>

Control	Data Type	Description	Example
Date Time Picker	Time	Sets or reads the selected date and time.	<code>DateTimePicker1.Value = "11/15/2006 2:55:12 PM";</code>
List Box	Integer	Reads the value of the selected item, or selects the item with that value if it exists.	<code>ListBox1.Value = "John Smith";</code>

## Configuring Radio Button Group Controls

You can use a Radio Button Group control element to exclusively select an option from a group of options at run time.

You can set the:

- 3D appearance of buttons.
- Layout of the radio button group options.

You can also use properties that are specific to the Radio Button Group control in scripting. At run time you can access the script to view and modify the Radio Button Group control.

## Setting the 3D appearance of a Radio Button Group Control

You can set the 3D appearance of a radio button group control. This affects how the option circles appear.

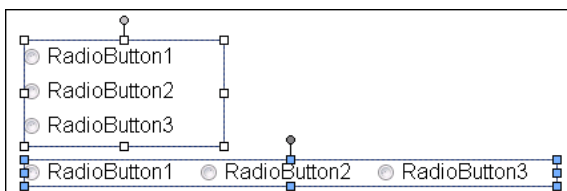
- Three-dimensional appearance
- Flat appearance in same color as option text

### To set the 3D appearance of a radio button group control

1. Select the radio button group control.
2. In the Properties Editor, select from the list for the ControlStyle property:
  - Click **ThreeD** for a three-dimensional appearance.
  - Click **Flat** for a flat two-dimensional appearance in the same color as the option text.

## Setting the Layout of the Radio Button Group Options

You can set the layout of the radio button group options in a vertical or horizontal direction.



### To set the layout of the radio button group options

1. Select the radio button group control.

2. In the Properties Editor, select from the list for the Layout property:
  - Click **Vertical** to arrange the options under each other.
  - Click **Horizontal** to arrange the options next to each other.

---

**Note:** You can set this option also in the radio button group animation dialog box.

---

## Using Radio Button Group-Specific Properties at Run Time

You can use properties that are specific to the Radio Button Group control at run-time. These properties are:

- **Count** - returns the number of radio buttons in the Radio Button Group control.
- **SelectedValue** - reads the value of the selected item, or selects the item with that value if it exists.

These properties are available when you browse for a Radio Button Group control in your HMI's attribute/tag browser.

## Configuring Check Box Controls

You can use a Check Box control for users to reset a Boolean attribute during run time.

You can set the following properties of the Check Box control:

- Default state, checked or unchecked.
- Caption text of the Check Box control button.
- 3D appearance of the Check Box control button.

### Setting the Default State of a Check Box Control

You can set the default state of a check box control to be checked or unchecked.

#### To set the default state of a check box control

1. Select the Check Box control.
2. In the Properties Editor, select from the list for the **Checked** property:
  - Click **False** to use an unchecked check box by default.
  - Click **True** to use a checked check box by default.

### Setting the Caption Text of a Check Box Control

You can set the caption text of a Check Box control.

#### To set the caption text of a Check Box control

1. Select the Check Box control.
2. In the Properties Editor, type a text string in the Caption property value box.

### Setting the 3D appearance of a Check Box Control

You can set the appearance of the check box within the control to be either flat or three-dimensional.



Three-dimensional appearance



Flat appearance in same color as caption text

### To set the 3D appearance of a Check Box control

1. Select the check box control.
2. In the Properties Editor, select from the list for the ControlStyle property:
  - Click **ThreeD** for a three-dimensional check box.
  - Click **Flat** for a flat two-dimensional check box in the same color as the caption text.

## Configuring Edit Box Controls

You can use an Edit Box control to create a box during run time in which users can enter text or view text.

You can configure the following properties of an Edit Box control:

- Set the default text.
- Wrap text to the next line in the edit box at design time and run time.
- Configure it so that the run-time text is read-only.

### Setting the Default Text in an Edit Box Control

You can set the default text that appears in an edit box control during run time.

#### To set the default text in an Edit Box control

1. Select the edit box control.
2. In the Properties Editor, type a text in the Text property. The text appears in the edit box control at design time. At run time, it can be overwritten with the value of a configured attribute.

### Configuring the Text to Wrap in an Edit Box Control

You can configure the edit box control to wrap text at design time and run time. This lets you view and type strings in a more compact way.

#### To configure text-wrapping in an edit box control

1. Select the edit box control.
2. In the Properties Editor, select from the list for the Multiline property:
  - Click **True** to enable text-wrapping at run time.
  - Click **False** to disable text-wrapping at run time.

### Configuring the Text to be Read-Only in an Edit Box Control

You can configure the Edit Box control to only show text at run time and prevent the run-time user from writing back to the associated attribute.

#### To configure the text to be read-only in an Edit Box control

1. Select the edit box control.
2. In the Properties Editor, set the ReadOnly property to **True**.

---

**Note:** To enable writing back to the associated attribute at run time, you can set the **ReadOnly** property to **False**.

---



## Configuring Combo Box Controls

You can use Combo Box controls to select an option from a foldable list.



You can configure:

- Drop-down type of combo box control.
- Width of the drop-down list.
- Integral height flag of the drop-down list to avoid clipping of the items in simple combo box controls.
- Maximum number of items to appear in the drop-down list.

You can also use properties that are specific to the Combo Box control in scripting. At run time, you can access the script to view and modify the items in the Combo Box control.

## Setting the Type of Combo Box Control

You can use one of the following combo box control types:

- Simple - no drop-down list, values can be entered
- DropDown - has a drop-down list, values can be entered
- DropDownList - has a drop-down list, values cannot be entered

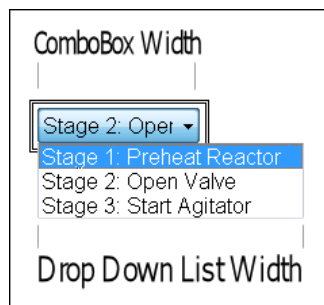
### To set the type of Combo Box control

1. Select the combo box control.
2. In the Properties Editor, select from the list for the DropDownType property:
  - Simple
  - DropDown
  - DropDownList

## Setting the Width of the Drop-Down List

You can set the width of the expanded drop-down list when the user clicks on it. This setting can be used to save space of the folded combo box control at run time.

Typically you set the drop-down list width greater than the width of the combo box on the canvas.



If you set the drop-down list width smaller than the combo box control width on the canvas, the drop-down list is the same width as the combo box control.

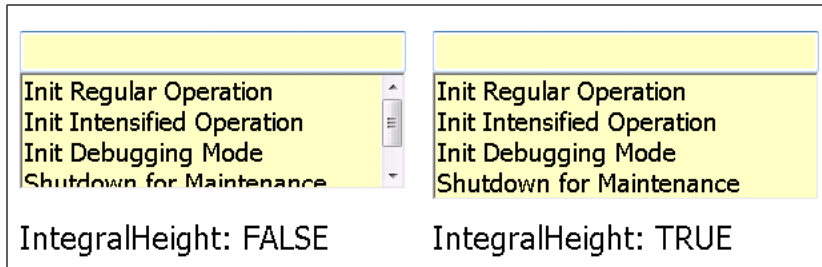
### To set the width of the combo box drop-down list

1. Select the combo box control.

- In the Properties Editor, type a width value in the DropDownWidth property value box.

## Avoiding Clipping of Items in the Simple Combo Box Control

You can avoid clipping of items in the simple combo box control list by setting the IntegralHeight property to true. The combo box list height is then changed to ensure that no items appear clipped.



### To avoid clipping of items in the drop-down list

- Select the combo box control.
- In the Properties Editor, select **True** as the value for the IntegralHeight property.

## Setting the Maximum Number of Items to Appear in the Combo Box Drop-Down List

You can limit the number of items that appear at any given time in the combo box drop-down list.

### To set the maximum number of items to appear in the drop-down list

- Select the combo box control.
- In the Properties Editor, type the maximum number as a value for the **MaxDropDownItems** property.

## Using Combo Box-Specific Properties at Run Time

You can use properties that are specific to the Combo Box control at run time.

- The count property returns the number of items in a Combo Box control.
- The **NewIndex** property returns the index of the last item added to the Combo Box list.

These properties are available when you browse for a Combo Box control in your HMI's attribute/tag browser.

## Configuring Calendar Controls

You can use the Calendar control to select a date from one or more monthly calendar sheets.

You can:

- Set the number of calendar month sheets to be shown.
- Set the first day of the week.
- Show or hide today's date on the bottom of the control.
- Set the fill and text colors of the calendar title.
- Set the text color for trailing dates.
- Set the date value of the Calendar Control that is used as default at run time.

## Setting the Number of Calendar Month Sheets

You can set the number of calendar month sheets to be shown by specifying the number of month columns and month rows. The number of columns and rows in the calendar control depends on the font size and the width of the calendar control.

For example, you can show six months in a calendar control by specifying two columns and three rows.

January, 2007						February, 2007								
Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	
31	1	2	3	4	5	6	4	5	6	7	8	1	2	3
7	8	9	10	11	12	13	11	12	13	14	15	16	17	
14	15	16	17	18	19	20	18	19	20	21	22	23	24	
21	22	23	24	25	26	27	25	26	27	28				
28	29	30	31											

March, 2007						April, 2007							
Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat
				1	2	3	1	2	3	4	5	6	7
4	5	6	7	8	9	10	8	9	10	11	12	13	14
11	12	13	14	15	16	17	15	16	17	18	19	20	21
18	19	20	21	22	23	24	22	23	24	25	26	27	28
25	26	27	28	29	30	31	29	30					

May, 2007						June, 2007							
Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat
		1	2	3	4	5	3	4	5	6	7	8	9
6	7	8	9	10	11	12	10	11	12	13	14	15	16
13	14	15	16	17	18	19	17	18	19	20	21	22	23
20	21	22	23	24	25	26	24	25	26	27	28	29	30
27	28	29	30	31			1	2	3	4	5	6	7

Today: 1/20/2007

### To set the number of calendar month sheets

1. Select the Calendar control.
2. In the **Properties Editor**, configure the calendar properties:
  - For the **CalendarColumns** property, specify the number of columns in the calendar control.
  - For the **CalendarRows** property, specify the number of rows in the calendar control.

## Setting the First Day of the Week

You can set the first day of the week for the calendar control. This is the day that appears on the most left column of each calendar month sheet.

You can set it to:

- The default as defined by the operating system.
- Any day of the week.

### To set the first day of the week

1. Select the Calendar control.
2. In the **Properties Editor**, select from the list for the **FirstDayOfWeek** property:
  - Click **Default** to use the operating system setting.
  - Click the day of the week.

## Showing or Hiding Today's Date on a Calendar Control

You can show or hide today's date on the bottom of a calendar control

### To show or hide today's date on the bottom of a calendar control

1. Select the Calendar control.
2. In the Properties Editor, set the ShowToday property to one of the following:
  - **True** to show today's date
  - **False** to hide today's date

## Setting Title Fill Color and Text Color on a Calendar Control

You can set the title fill color and title text color on a calendar control.

Changing the title fill color also affects the:

- Color of the week days.
- Fill color of the indication box of today's date.

When you change the title text color, this also affects the text color of the indication box of today's date.



### To change the title fill color of a Calendar control

1. Select the Calendar control.
2. In the **Properties Editor**, click the browse button for the **TitleFillColor** property. The **Select FillColor** dialog box appears. For more information, see *Setting Style* on page 103.
3. Select a color and click **OK**. The title fill color is changed accordingly.

### To change the title text color of a Calendar control

1. Select the Calendar control.
2. In the **Properties Editor**, click the browse button for the **TitleTextColor** property. The **Select FillColor** dialog box appears. For more information, see *Setting Style* on page 103.
3. Select a color and click **OK**. The title text color is changed accordingly.

## Setting the Text Color for Trailing Dates in a Calendar Control

You can set the text color for dates outside the month for any month sheet in a calendar control.

### To set the text color for trailing dates

1. Select the Calendar control.
2. In the **Properties Editor**, click the browse button for the **TrailingTextColor** property. The **Select Text Color** dialog box appears. For more information, see *Setting Style* on page 103.
3. Select a color and click **OK**. The text color of the trailing dates is changed accordingly.

## Setting the Default Value of the Calendar Control

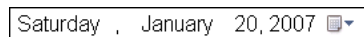
You can set the default value of the Calendar Control. The default value is a date that the control uses when it is shown the first time.

### To set the default value of the calendar control

1. Select the Calendar control.
2. In the **Properties Editor**, set the **DefaultValue** property to the date value you want to use as default at run time.

## Configuring DateTime Picker Controls

Use the DateTime Picker control to select a date or time.



You can configure the DateTime Picker control to show:

- A long format, such as Friday, August 11, 2008.
- A short format, such as 8/11/2008.
- Just the time, such as 9:16:36 PM.
- A custom time format, such as 8/11/2008 9:16:36 PM.

You can also set the default value of the DateTime Picker control.

### To set the long date format

1. Select the DateTime Picker control.
2. In the Properties Editor, set the Format property to **Long**.

### To set the short date format

1. Select the DateTime Picker control.
2. In the Properties Editor, set the Format property to **Short**.

### To set only time display

1. Select the DateTime Picker control.
2. In the Properties Editor, set the Format property to **Time**.

### To set a custom date/time format

1. Select the DateTime Picker control.
2. In the Properties Editor, set the Format property to **Custom**.
3. Type the time format in the value box for the **CustomFormat** property. Use the following letters as placeholders:

- |    |                                                                                   |
|----|-----------------------------------------------------------------------------------|
| h  | The one or two-digit hour in 12-hour format.                                      |
| hh | The two-digit hour in 12-hour format. Single digit values are preceded by a zero. |
| H  | The one or two-digit hour in 24-hour format.                                      |
| HH | The two-digit hour in 24-hour format. Single digit values are preceded by a zero. |

t	The one-letter AM/PM abbreviation ("AM" is shown as "A").
tt	The two-letter AM/PM abbreviation ("AM" is shown as "AM").
m	The one or two-digit minute.
mm	The two-digit minute. Single digit values are preceded by a zero.
s	The one or two-digit seconds.
ss	The two-digit seconds. Single digit values are preceded by a zero.
d	The one or two-digit day.
dd	The two-digit day. Single digit day values are preceded by a zero.
ddd	The three-character day-of-week abbreviation.
dddd	The full day-of-week name.
M	The one or two-digit month number.
MM	The two-digit month number. Single digit values are preceded by a zero.
MMM	The three-character month abbreviation.
MMMM	The full month name.
y	The one-digit year (2001 is shown as "1").
yy	The last two digits of the year (2001 is shown as "01").
yyyy	The full year (2001 is shown as "2001").

You can use any other characters, except "g" in the property. These characters then appear at design time and run time in the control.

### To set the default value in a DateTime Picker control

1. Select the DateTime Picker control.
2. In the Properties Editor, set the DefaultValue property to the date and time value you want to use as default at run time.

## Configuring List Box Controls

You can create a list box for users to select an option from a scrollable list during run time.

You can:

- Configure the list box to avoid clipping of its contained items. When you set the Integral Height flag, the list box control is resized so that no items are clipped.
- Specify if you want the control to be scrollable in horizontal direction at run time. This enables the user to see the full text if the item captions are wider than the control itself.
- Use properties that are specific to the List Box control in scripting. At run time you can access the script to view and modify the items in the List Box control.

## Avoiding Clipping of Items in the List Box Control List

In the list of a List Box control, some items may appear vertically clipped. You can configure the List Box control to avoid this clipping by setting the `IntegralHeight` property.

### To avoid clipping of items in the List Box control

1. Select the list box control.
2. In the **Properties Editor**, select **True** as value for the **IntegralHeight** property.

## Using a Horizontal Scroll Bar in a List Box Control

You can configure a horizontal scroll bar in a List Box Control so that at run time the user can scroll the list horizontally to see items that are wider than the control.

### To configure a horizontal scroll bar

1. Select the List Box control.
2. In the **Properties Editor**, select **True** as value for the **HorizontalScrollbar** property.

## Using List Box-Specific Properties at Run Time

You can use properties that are specific to the List Box control at run time.

- The **Count** property returns the number of items in a List Box control.
- The **NewIndex** property returns the index of the last item added to the List Box list.
- The **SelectedValue** property reads the value of the selected item, or selects the item with that value if it exists.
- The **TopIndex** property returns the index of the top most item in the list.

These properties are available when you browse for a List Box control in your HMI's attribute/tag browser.



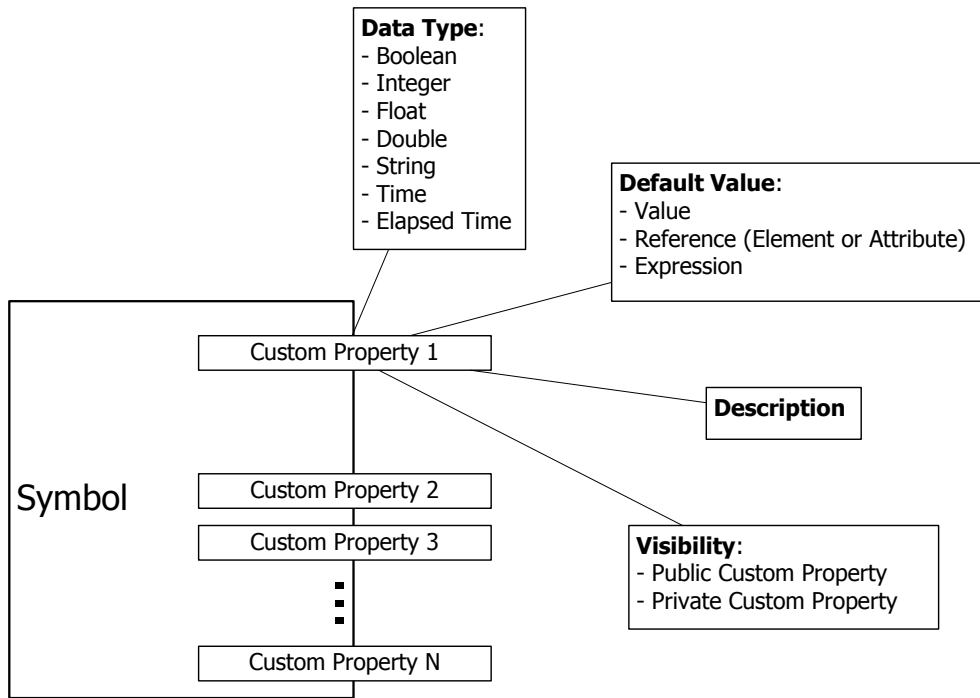


# CHAPTER 7

## Using Custom Properties

### About Custom Properties

You can configure and use custom properties to extend the functionality of symbols and use them in combination with the HMI/SCADA software attributes or tags. You can use binding with custom properties to dynamically change the reference of a custom property.



You can associate custom properties with functionality you want exposed and that you want to be reusable. You can also use custom properties to connect an embedded graphic to attributes and tags in your HMI/SCADA software.

### Managing Custom Properties

You manage all custom properties of a graphic using the **Edit Custom Properties** dialog box.

From the **Custom Properties** dialog box, you can:

- Add and delete custom properties.
- Set the types and data types of custom properties.
- Set the default values of custom properties.
- Determine the visibility of each custom property.
- Add a description for each custom property.
- Validate and clear custom properties.

You can also:

- Rename custom properties.
- Link custom properties to external sources.
- Override custom properties with new values.
- Revert custom property values to their default values.

## Adding and Deleting Custom Properties

You can add and delete custom properties from a graphic.

### To add a custom property

1. Click the canvas to cancel any selected elements.
2. On the **Special** menu, click **Custom Properties**. The **Edit Custom Properties** dialog box appears.
3. Click the **Add** icon. A new line is added in the custom properties list.
4. Type a name for the new custom property and click **Enter**.

You can see the name of the graphic and the custom property in the header of the dialog box.

- If the graphic includes an embedded graphic, the name of the custom property cannot be the same as the name of the embedded graphic or of an element of the embedded graphic.
  - If the graphic includes a script, the name of the custom property and a nested class property in the script cannot be the same.
5. Configure the custom property on the right side of the **Edit Custom Properties** dialog box. For more information, see *Configuring Custom Properties* on page 146.
  6. Click **OK**.

### To delete a custom property

1. Click the canvas to cancel any selected elements.
2. On the **Special** menu, click **Custom Properties**. The **Edit Custom Properties** dialog box appears.
3. Select the custom property you want to delete and click the **Remove** icon. When a message appears requesting confirmation to delete the custom property, click **Yes**. The custom property is removed from the custom properties list.
4. Click **OK**.

## Configuring Custom Properties

You can configure custom properties when you create them or at a later point of time.

### To configure a custom property

1. Click the canvas of the graphic.
2. On the **Special** menu, click **Custom Properties**. The **Edit Custom Properties** dialog box appears.
3. Select the custom property you want to edit. The configuration for the selected custom property appears at the right of the dialog box.








---

**Note:** The header of the configuration area shows the graphic name on the right, for example "Symbol\_001," and it shows the custom property name on the left, for example "MyCustomProperty." Its reference in a script would be "Symbol\_001.MyCustomProperty."

---

4. In the **Data Type** list, click the data type of the custom property. You can select one of the following:

Data Type	Graphic
-----------	---------

Boolean	
Double	
Elapsed Time	
Float	
Integer	
String	
Time	

5. If you want to:
  - Make the property read-only at design time and prevent further changes to it when the graphic is embedded into another graphic, click the Lock icon.
  - Make the property read-only at run time and prevent its value being changed, click the Lock icon.
6. In the **Default Value** box, type a literal value, reference, or expression or browse for a reference using the **Browse** icon.
7. If the selected data type is String, Time or Elapsed Time, you can click the **T** icon or tag icon.
  - Select the **T** icon to indicate that the default value is a static value.
  - Select the tag icon to indicate that the default value is a reference to a value.
8. In the **Visibility** box, configure how the graphic is visible. Do one of the following:
  - Click **Public** if you want the custom property to be visible and can be used in a source graphic if the graphic is embedded.
  - Click **Private** if you want the custom property to be hidden and no reference be made to it outside of the defining graphic.
9. In the **Description** box, type a meaningful description for the custom property.

## Validating Custom Properties

You can validate custom properties to track down and avoid configuration errors.

### To validate a custom property

1. Click on the canvas to cancel any selected elements.
2. On the **Special** menu, click **Custom Properties**. The **Edit Custom Properties** dialog box appears.
3. Select the custom property you want to validate and click the **Validate** icon. Required boxes are highlighted by a red box, possible errors appear in the status area under the custom properties list.

## Clearing the Configuration of Custom Properties

You can clear the configuration of custom properties. This resets the properties to their default values.

### To clear the configuration of a custom property

1. In the **Edit Custom Properties** dialog box, select the custom property.
2. Click the **Clear** icon. The configured values are reset to their default values.

## Renaming Custom Properties

You can rename custom properties.

### To rename a custom property

1. In the **Edit Custom Properties** dialog box, select the custom property.
2. Click the custom property again. The custom property is in edit mode.
3. Type the new custom property name and click **Enter**. The custom property is renamed.

---

#### Notes:

If the graphic includes an embedded graphic, the name of the custom property cannot be the same as the name of the embedded graphic or of an element of the embedded graphic.

If the graphic includes a script, the name of the custom property and a nested class property in the script cannot be the same.

---

## Linking Custom Properties to External Sources

You can link custom properties of a graphic directly to external sources by:

- Configuring objects that point to external sources and then point the custom property at the corresponding attribute reference.
- Configuring a special reference syntax to your HMI/SCADA software in the **Default Value** box. When you embed the graphic into a window in your HMI/SCADA software, the referenced tags connect to the tags of your HMI/SCADA software.

## Overriding Custom Properties

You can override the custom property default values of embedded graphics within graphics in the Industrial Graphic Editor or your HMI/SCADA software.

---

**Note:** When you override the custom property, it appears bold in the custom property list.

---

You can override the following custom property values:

- Default value
- Visibility, but only from public to private, not private to public
- Description
- Locked state
- String mode setting

You cannot override the data type of a custom property.

## Reverting to Original Custom Property Values

After you override a custom property value, you can revert to the original custom property value. This can be done for overridden custom properties of embedded graphics in other graphics and in your HMI/SCADA software.

### To revert to the original custom property value

- In the **Edit Custom Properties** dialog box, click the Revert icon. The custom property value reverts to its original value.





# CHAPTER 8

## Animating Graphic Elements

### About Animations

You can use animations to change the appearance of graphic elements at run time. Animations are driven by data that comes from attribute or tag values and expressions as well as element properties.

You can use:

- **Visualization animations** such as visibility, fill style, line style, text style, blinking, percent fill horizontal, percent fill vertical, horizontal location, vertical location, width, height, orientation, value display or tooltip.
- **Interaction animations** such as disable, user input, horizontal slider, vertical slider, pushbutton, action script, show graphic or hide graphic.
- **Element-specific animations** for the Status element and Windows common control elements.

Each element in your Industrial graphic can have one or more animations. You can disable and enable individual animations. You can also cut, copy and paste animations between elements. Only animations supported by the target element are pasted.

You can also substitute references and strings in animations.

---

**Note:** Not all animations are available for all element types. Some animations do not make logical sense, such as line style with a text element. You cannot select or copy these invalid combinations.

---

### Adding an Animation to an Element

You can add one or more animations to a single element in your Industrial graphic.

#### To add an animation to an element

1. Select the element to which you want to add an animation.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.  
You can also add an animation from the Animation Summary in the lower right corner of the Industrial Graphic Editor.
3. Click the **Add** icon. The list of animations appears.
4. Select an animation from the list. The animation is added to the Animation list. You can configure the selected animation from the **Edit Animations** dialog box.

---

**Note:** Depending on the animation type, you may get an animation state selection panel instead. For more information, see *Reviewing which Animations are Assigned to an Element* on page 151.

---

### Reviewing which Animations are Assigned to an Element

You can review which animations are assigned to an element and change the number of animations or their configuration at the same time.

### To review which animations are assigned to an element

1. Select the element. The assigned animations appear in the Animation Summary in the lower right of the Industrial Graphic Editor.

You can also review which animations are assigned to an element by double-clicking it.

2. Select an animation to view further information on how the element is configured with that animation.

## Showing and Hiding the Animation List

You can show or hide the Animation list. If you hide the Animation list, the configuration space expands, giving you more space to configure the animations.

### To hide the Animation list

- In the **Edit Animations** dialog box, click the **Hide** icon. The Animation list hides and the configuration space expands.

### To show the Animation list

- In the **Edit Animations** dialog box, click the **Show** icon. The Animation list appears and the configuration space reduces to its default width.

## Removing Animations from an Element

You can remove an animation from an element by using the **Edit Animations** dialog box. You can remove animations from an element for:

- Individual animations
- All animations at the same time

### To remove an animation from an element

1. Select the element in which you want to remove an animation.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.

You can also remove an animation from the Animation Summary in the lower right of the Industrial Graphic Editor.

3. Select the animation you want to remove from the Animation list.
4. Click the **Remove** icon. A message appears.
5. Click **Yes**. The animation is removed from the list and no longer associated with the element.

### To remove all animations from an element

1. Select one or more elements from which you want to remove all animations.
2. Do one of the following:
  - Right-click, point to **Animations** and then click **Clear**.
  - On the **Edit** menu, point to **Animations**, and then click **Clear**.

All animations are removed from the selected elements.

## Enabling and Disabling Animations

You can enable or disable animations for an element. When you disable an animation, its configuration is not lost. This lets you see, for example, each animation independently from each other.

### To disable an animation

1. Select the element with the animation you want to disable.



2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.  
You can also disable animations from the Animation Summary in the lower right corner of the Industrial Graphic Editor.
3. Locate the animation you want to disable from the Animation list on the left of the dialog box.
4. Select **Disabled** from the list of that row.
5. Repeat for any other animations you want to disable and click **OK** when you are done.

#### To enable an animation

1. Select the element with the animation you want to enable.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.  
You can also enable animations from the Animation Summary of the Industrial Graphic Editor.
3. Locate the animation you want to enable from the Animation list.
4. Select **Enabled** from the list of that row.
5. Repeat for any other animations you want to enable and click **OK** when you are done.

## Validating the Configuration of an Animation

You can validate the configuration of an animation. If the configuration contains an error, an exclamation mark appears next to the **Animation** icon.

Examples of animation configuration errors include:

- Animation is disabled
- Syntax errors such as data mismatches
- Required values not specified
- Specified values out of valid range

#### To validate the configuration of an animation

1. Select the element that contains the animations you want to validate.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Select the animation you want to validate.
4. Click the **Validate** icon. The currently selected animation is validated. Possible errors are highlighted.

## Clearing the Configuration from an Animation

You can clear all data from the configuration boxes of an animation and reset the settings to their defaults.

#### To clear all data from the configuration boxes of an animation

1. In the **Edit Animations** dialog box, select the animation.
2. In the configuration panel, click the **Clear** icon. All data from the configuration boxes is cleared and the settings are reset to their defaults.

## Managing Animations

You can easily manage animations in the **Edit Animations** dialog box. You can:

- Change the way the list of animations appears.

- Switch easily between multiple animations of an element.

You can also do this for the Animation Summary in the lower right corner of the Industrial Graphic Editor.

## Organizing the Animation List

You can organize the list of animations alphabetically or by category.

### To organize the Animation list

- In the **Edit Animations** dialog box, click the:
  - **Alphabetic sort** icon to sort alphabetically.
  - **Category** icon to sort by category.

## Switching between Animations

If you configure more than one animation for an element, you can easily switch between their configuration panels without having to use the Animation list. This is particularly useful when the Animation list is hidden.

### To switch between animations

- In the **Edit Animations** dialog box, on the configuration panel click the left or right arrow icon.  
The configuration panel changes to the configuration panel of the previous or next animation.

## Configuring Common Types of Animations

Every animation type has its own set of configuration parameters. This section shows you how to configure each type of animation and what references it can use.

You can configure:

- Visualization animations such as:
  - Visibility animations
  - Fill style, line style or text style animations
  - Blink animations
  - Alarm Border animations
  - Horizontal or vertical percent fill animations
  - Horizontal or vertical location animations
  - Width or height animations
  - Point animations
  - Orientation animations
  - Value display animations
  - Tooltip animations
- Interaction animations such as:
  - Disable animation
  - User input animation
  - Horizontal and vertical slider animations
  - Pushbutton animations

- Action script animations
- Show or hide animations

## Configuring a Visibility Animation

You can configure an element with a visibility animation.

### To configure an element with a visibility animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Visibility**. The visibility animation is added to the Animation list and the **Visibility** configuration panel appears.
4. In the **Boolean** box, type a Boolean numeric value, attribute reference or expression.
5. Select **True, 1, On** if you want the element to show, when the expression is true, otherwise select **False, 0, Off**.

## Configuring a Fill Style Animation

You can configure an element with a:

- Boolean fill style animation.
- Truth table fill style animation.

The truth table fill style animation lets you:

- Associate expressions of any data type supported by the HMI with a fill style.
- Define as many fill styles as you require and associate each one with a condition.

You can define the conditions by specifying an comparison operator (=, >, >=, <, <=) and a breakpoint, which itself can be a value, an attribute reference, or an expression.

You can add conditions, delete conditions, and also change the order in which the conditions are processed.

## Configuring a Boolean Fill Style Animation

You can configure an element with a discrete fill style animation.

### To configure an element with a Boolean fill style animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Fill Style**. The fill style animation is added to the Animation list and the **Fill Style** state selection panel appears.
4. Click the **Boolean** button. The **Boolean Fill Style** configuration panel appears.
5. In the **Boolean** box, type a Boolean numeric value, attribute reference or expression.
6. Clear **Color** in the **True, 1, On** area or **False, 0, Off** area if you do not want a different fill style for the true or false condition than the default fill style.
7. In the **True, 1, On** area, click the color box to configure the fill color when the expression is true. The **Select FillColor** dialog box appears. For more information, see *Setting Style* on page 103.
8. In the **False, 0, Off** area, click the color box to configure the fill color when the expression is false. The **Select FillColor** dialog box appears. For more information, see *Setting Style* on page 103.

9. Click **OK**.

#### To set default fill style in a Boolean fill style animation

1. Open the **Edit Animations** dialog box, **Boolean Fill Style** panel.
2. In the **Element Fill Style** area, click the color box to select a style from the **Select FillColor** dialog box.

#### To use default fill style in a Boolean fill style animation

1. Open the **Edit Animations** dialog box, **Boolean Fill Style** panel.
2. Clear **Color** to use the corresponding default fill style.

## Configuring a Truth Table Fill Style Animation

You can configure an element with a fill style animation based on a truth table.

#### To configure an element with a truth table fill style animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Fill Style**. The fill style animation is added to the Animation list and the **Fill Style** state selection panel appears.
4. Click the **Truth Table** button. The **Truth Table Fill Style** configuration panel appears.
5. In the **Expression Or Reference** area:
  - Select the data type of the expression from the list.
  - Type a value, attribute reference or expression in the text box.
6. If the data type of the expression is string or internationalized string, you can specify to ignore the case by selecting **Ignore Case**.
7. In the **Truth Table**, click the color box in the **Color** column. The **Select FillColor** dialog box appears. For more information, see *Setting Style* on page 103.
8. In the **Operator** column, select the comparison operator.
9. In the **Value or Expression** column, type a value, attribute reference, or expression.
10. To add further conditions, see *To add a condition to a truth table fill style animation*.
11. Click **OK**.

#### To set the default fill style for a truth table fill style animation

1. Open the **Edit Animations** dialog box, **Truth Table Fill Style** panel.
2. In the **Element Fill Style** area, click the color box. The **Select FillColor** dialog box appears. For more information, see *Setting Style* on page 103.

#### To use the default fill style in a truth table fill style animation

1. Open the **Edit Animations** dialog box, **Truth Table Fill Style** panel.
2. Locate the condition for which you want to set the style to default style.
3. Clear the mark for that condition in the **Color** column of the truth table. The associated style is the same as the style for the **Element Fill Style**.

#### To add a condition to a truth table fill style animation

1. Open the **Edit Animations** dialog box, **Truth Table Fill Style** panel.
2. Click the **Add** icon. An additional condition is added to the truth table.

3. Configure color, operator and breakpoint value according to your requirements.

### To delete a condition from an analog fill style animation

1. Open the **Edit Animations** dialog box, **Truth Table Fill Style** panel.
2. Select the condition you want to delete.
3. Click the **Remove** icon. The condition is removed.

### To change the processing order of fill style conditions

1. Open the **Edit Animations** dialog box, **Truth Table Fill Style** panel.
2. Select the condition you want to move up or down the condition list in order for it to be processed sooner or later.
3. Click the:
  - **Arrow up** icon to move the condition up in the truth table.
  - **Arrow down** icon to move the condition down in the truth table.

For example, you want to model an analog fill color animation that describes the following conditions:

- When the attribute TankLevel\_001.PV is 0 then the fill style is solid black.
- When the attribute TankLevel\_001.PV is smaller than 20, then the fill style is solid red.
- When the attribute TankLevel\_001.PV is greater than the attribute Standards.TankMax then the fill style is red with a diagonal pattern.
- In all other cases, the fill style is solid blue.

## Configuring a Line Style Animation

You can configure an element with a:

- Boolean line style animation.
- Truth table line style animation.

The truth table line style animation lets you:

- Associate expressions of any data type supported by your HMI/SCADA software with a line style.
- Define as many line styles as you want and associate each one with a condition.

You can define the conditions by specifying an comparison operator (=, >, >=, <, <=) and a breakpoint, which itself can be a value, an attribute reference or an expression.

You can add conditions, delete conditions and also change the order in which the conditions are processed.

## Configuring a Boolean Line Style Animation

You can configure an element with a Boolean line style animation. You can use a new style or use all or parts of the default appearance of a line for:

- Line style.
- Line thickness.
- Line pattern.

### To configure an element with a Boolean line style animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.

3. Click the **Add** icon and select **Line Style**. The line style animation is added to the Animation list and the **Line Style** state selection panel appears.
4. Click the **Boolean** button. The **Boolean Line Style** configuration panel appears.
5. In the **Boolean** box, type a Boolean numeric value, attribute reference or expression.
6. In the **True, 1, On** area, click the **Color** box to configure the line style when the expression is true. The **Select FillColor** dialog box appears. For more information, see *Setting Style* on page 103.
7. In the **Weight** box, type a value for the line thickness when the expression is true.
8. From the **Pattern** list, select a line pattern for the line when the expression is true.
9. Repeat the above steps for the false condition in the **False, 0, Off** area.
10. Click **OK**.

#### **To set default line style, thickness and/or pattern in a Boolean line style animation**

1. Open the **Edit Animations** dialog box, **Boolean Line Style** panel.
2. In the **Element Line Style** area, select a style, type a value for the width and select a pattern for the default Boolean line style.

#### **To use default line style, thickness and/or pattern in a Boolean line style animation**

1. Open the **Edit Animations** dialog box, **Boolean Line Style** panel.
2. In the **True, 1, On** or **False, 0, Off** areas, clear **Color**, **Weight** and/or **Pattern** to use the corresponding default style, weight and/or pattern.

## **Configuring a Truth Table Line Style Animation**

You can configure an element with a truth table line style animation.

#### **To configure an element with a truth table line style animation**

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Line Style**. The line style animation is added to the Animation list and the **Line Style** state selection panel appears.
4. Click the **Truth Table** button. The **Truth Table Line Style** configuration panel appears.
5. In the **Expression or Reference** box:
  - Select the data type of the expression from the list.
  - Type a value, attribute reference or expression in the text box.
6. If the data type of the expression is string or internationalized string, you can specify to ignore the case by selecting **Ignore Case**.
7. In the **Truth Table**, click the color box in the **Color** column. The **Select FillColor** dialog box appears. For more information, see *Setting Style* on page 103.
8. Select the truth options. Do one of more of the following:
  - In the **Weight** column, type a value for the line weight.
  - In the **Pattern** column, select a line pattern.
  - In the **Operator** column, select the comparison operator.
  - In the **Value or Expression** column, type a value, attribute reference or expression.
  - To add further conditions, see *To add a condition to a truth table line style animation*.

9. Click **OK**.

### To set the default line style, width or pattern for a truth table line style animation

1. Open the **Edit Animations** dialog box, **Truth Table Line Style** panel.
2. In the **Element Line Style** area, select a style, type a value for the width and select a pattern for the default truth table line style.

### To use the default line style, width or pattern in a truth table line style animation

1. Open the **Edit Animations** dialog box, **Truth Table Line Style** panel.
2. Locate the condition for which you want to change the line style, width or pattern.
3. To use the default line style for the condition, clear the mark in the **Color** column of the truth table.
4. To use the default line width for the condition, clear the mark in the **Width** column of the truth table.
5. To use the default line pattern for the condition, clear the mark in the **Pattern** column of the truth table.

### To add a condition to a truth table line style animation

1. In the **Edit Animations** dialog box, **Truth Table Line Style** panel, click the **Add** icon. An additional condition is added to the truth table.
2. Configure color, weight, pattern, operator and breakpoint value according to your requirements.

### To delete a condition from an analog line color animation

1. In the **Edit Animations** dialog box, **Truth Table Line Style** panel, select the condition you want to delete.
2. Click the **Remove** button. The condition is removed.

### To change the processing order of line style conditions

1. Open the **Edit Animations** dialog box, **Truth Table Line Style** panel
2. Select the condition you want to move up or down the condition list in order for it to be processed sooner or later.
3. Click the:
  - **Arrow up** icon to move the condition up in the truth table.
  - **Arrow down** icon to move the condition down in the truth table.

## Configuring a Text Style Animation

You can configure an element with a:

- Boolean text style animation.
- Truth table text style animation.

The truth table text style animation lets you:

- Associate expressions of any data type supported by your HMI/SCADA software with a text style.
- Define as many text styles as you want and associate each one with a condition.

You can define the conditions by specifying a comparison operator (=, >, >=, <, <=) and a breakpoint, which itself can be a value, an attribute reference or an expression.

You can add conditions, delete conditions and also change the order in which the conditions are processed.

## Configuring a Boolean Text Style Animation

You can configure an element with a Boolean text style animation.

### To configure an element with a Boolean text style animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Text Style**. The text style animation is added to the Animation list and the **Text Style** state selection panel appears.
4. Click the **Boolean** button. The **Boolean Text Style** configuration panel appears.
5. In the **Boolean** box, type a Boolean numeric value, attribute reference or expression.
6. In the **True, 1, On** area, click the **Color** box to configure the text style when the expression is true. The **Select FillColor** dialog box appears. For more information, see *Setting Style* on page 103.
7. Click the browse button for the **Font** box, to select a font, font style and size for the text when the expression is true.
8. Repeat the above steps for the false condition in the **False, 0, Off** area.
9. Click **OK**.

### To set default text style and/or font in a Boolean text style animation

1. Open the **Edit Animations** dialog box, **Boolean Text Style** panel.
2. In the **Element Text Style** area, select a style and/or a font for the default Boolean text style.

### To use default text style and/or font in a Boolean text style animation

1. Open the **Edit Animations** dialog box, **Boolean Text Style** panel.
2. In the **True, 1, On** or **False, 0, Off** areas, clear **Color** and/or **Font** to use the corresponding default style and/or font.

## Configuring a Truth Table Text Style Animation

You can configure an element with a truth table text style animation.

### To configure an element with a truth table text style animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Text Style**. The text style animation is added to the Animation list. The **Text Style** information page appears.
4. Click the **Truth Table** button. The **Truth Table Text Style** configuration panel appears.
  - Select the data type of the expression from the list.
  - Type a value, attribute reference or expression in the text box.
5. If the data type of the expression is string or internationalized string, you can specify to ignore the case by selecting **Ignore Case**.
6. In the **Truth Table**, click the color box in the **Color** column. The **Select FillColor** dialog box appears. For more information, see *Setting Style* on page 103.
7. Select the truth options. Do one of more of the following:
  - Click on the cell in the **Font** column to select a font.
  - In the **Operator** column, select the comparison operator.



- In the **Value or Expression** column, type a value, attribute reference or expression.
- To add further conditions, see To add a condition to a truth table text style animation.

8. Click **OK**.

### To set the default text style or font for a truth table text style animation

1. Open the **Edit Animations** dialog box, **Truth Table Text Style** panel.
2. In the **Element Text Style** area, select a style and a font for the default truth table text style.

### To use the default text style or font in a truth table text style animation

1. Open the **Edit Animations** dialog box, **Truth Table Text Style** panel.
2. Locate the condition for which you want to change the text style or font.
3. To use the default text style for the condition, clear the mark in the **Color** column of the truth table.
4. To use the default font for the condition, clear the mark in the **Font** column of the truth table.

### To add a condition to a truth table text style animation

1. In the **Edit Animations** dialog box, **Truth Table Text Style** panel, click the **Add** icon. An additional condition is added to the truth table.
2. Configure style, font, operator and breakpoint value according to your requirements.

### To delete a condition from a truth table text style animation

1. In the **Edit Animations** dialog box, **Truth Table Text Style** panel, select the condition you want to delete.
2. Click the **Remove** button.

### To change the processing order of text style conditions

1. Open the **Edit Animations** dialog box, **Truth Table Text Style** panel
2. Select the condition you want to move up or down the condition list in order for it to be processed sooner or later.
3. Click the:
  - **Arrow up** icon to move the condition up in the truth table.
  - **Arrow down** icon to move the condition down in the truth table.

## Configuring a Blink Animation

You can configure an element with a blink animation. You can specify:

- The blinking speed: slow, medium or fast.
- If the element should blink invisibly or if it should blink with specified colors.

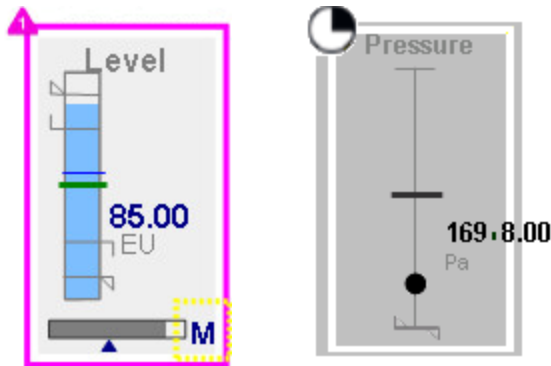
### To configure an element with a blink animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Blink**. The blink animation is added to the Animation list and the **Blink** configuration panel appears.
4. In the **Boolean** box, type a Boolean numeric value, attribute reference or expression.
5. In the **Blink When Expression Is** area, select:
  - **True, 1, On** to enable blinking when the expression is true.

- **False, 0, Off** to enable blinking when the expression is false.
6. In the **Blink Speed** area, select **Slow, Medium** or **Fast** for the blinking speed.
  7. In the **Blink Attributes** area, select **Blink Visible With These Attributes** or **Blink Invisible**.
  8. If you select **Blink Visible With These Attributes**, you can configure the styles used at run time for the text, line and fill component of the element. Click on the corresponding color box, and the **Select FillColor** dialog box appears. For more information, see *Setting Style* on page 103.
  9. Click **OK**.

## Configuring an Alarm Border Animation

Alarm Border animation shows a highly visible border around a graphic or graphic element when an alarm occurs. The color and fill pattern of the border indicates the severity and current state of the alarm, if supported by your HMI/SCADA software. Plant operators can quickly recognize alarm conditions when Alarm Border animation is used



Alarm Border animation also shows an indicator icon at the top left corner of the border around a closed graphic element. For open pie or arc graphic elements, the indicator icon is placed at the top-left most location of the start and end points.

Alarm severity (1-4) or the current alarm mode (Shelved, Silenced, Disabled) appear as part of the indicator icon. The indicator icon can be shown or hidden as a configurable option of Alarm Border animation.

Alarm Border animation adheres to the following precedence rules with other functions that can change the appearance of a graphic:

1. Quality status
2. Alarm Border animation
3. Element Style animation
4. Style animations
5. Element Style on canvas

## Understanding Requirements of Alarm Border Animations

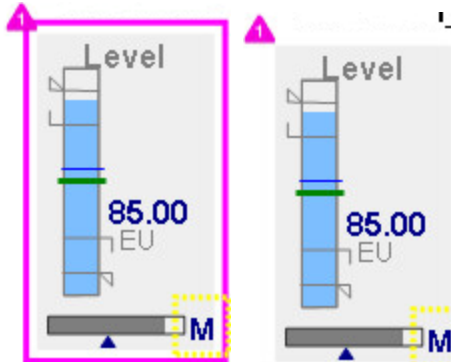
Alarm border animation can be applied to all types of graphics except embedded graphics and nested groups. Alarm border animation can also be applied to graphic elements and group elements.

## Understanding the Behavior of Alarm Border Animations

Alarm Border animation appears around a graphic element based on the current state of the object's aggregated alarm attributes, or other configured alarm inputs supported by your HMI/SCADA software. The appearance of the alarm border itself reflects the current alarm state and the user's interaction with the alarm.

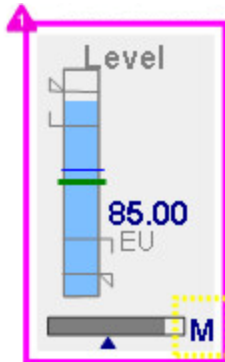
- A graphic's process value transition into an alarm state.

Alarm Border animation appears around the graphic based on alarm severity with blinking.



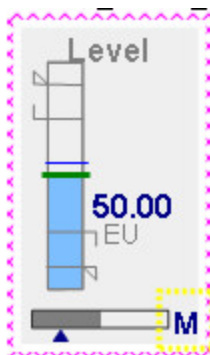
- The user acknowledges an alarm with the process value still in an alarm state.

Alarm Border animation appears around the graphic without blinking.



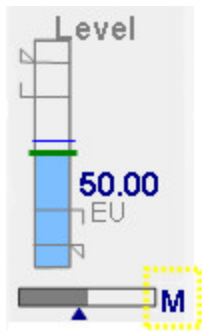
- The alarm value returns to normal without the user acknowledging the alarm.

Alarm Border animation remains around the graphic in a defined Return to Normal visual style without blinking.



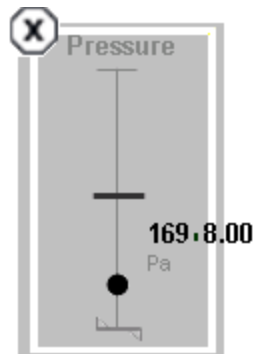
- The alarm value returns to normal and the user acknowledges the alarm.

Alarm Border animation no longer appears around a graphic.



- The user suppresses an alarm.

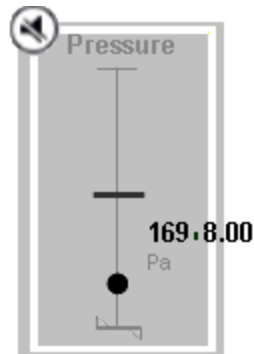
Alarm Border animation remains around the graphic in a defined suppressed/disabled visual style without blinking. The indicator shows the suppressed/disabled alarm mode icon.



- The user silences an alarm

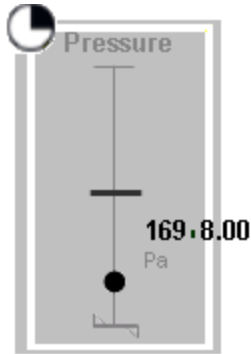
Alarm Border animation remains around the graphic in a defined silenced visual style without blinking. The indicator shows the silenced alarm mode icon.

A silenced alarm mode takes precedence over an alarm in shelved mode.



- The user shelves an alarm

Alarm Border animation remains around the graphic in a defined shelved visual style without blinking. The indicator shows the shelved alarm mode icon.



If a new alarm condition occurs when Alarm Border animation appears around a graphic, the animation updates to show the new alarm state. In the case of aggregation alarms, Alarm Border animation shows the highest current alarm state.

## Configuring Alarm Border Animation

Alarm Border animation can be configured by selecting **Alarm Border** from the list of **Visualization** animations. For Situational Awareness Library symbols, Alarm Border animation can be selected as a Wizard Option of the graphic.

The **Alarm Border** animation dialog box contains mutually exclusive fields to set the referenced attributes for aggregate or individual alarms.

If your HMI/SCADA software supports aggregating alarms from different plant or functional areas, you can specify Alarm Border animation by entering an attribute, object, or tag name in the **Use Standard Alarm-Urgency References** field of the **Alarm Border** dialog box.

The selected object attributes or tags typically map to the following aggregation alarm attributes:

- AlarmMostUrgentAcked
- AlarmMostUrgentInAlarm
- AlarmMostUrgentMode
- AlarmMostUrgentSeverity
- AlarmMostUrgenShelved

For individual alarms, you can specify Alarm Border animation by entering attribute or object names in the **Use Custom Alarm-Urgency References** fields of the **Alarm Border** dialog box.

- **InAlarm Source**

Indicates the InAlarm status (True/False) of the most urgent alarm that is in the InAlarm state or waiting to be Acked. If no alarms are in the InAlarm state or waiting to be Acked, the value is False.

- **Acked Source**

Indicates the acknowledgement status (True/False) of the most urgent alarm that is in the InAlarm state or waiting to be Acked. If no alarms are in an InAlarm state or waiting to be Acked, the value is True, which means no acknowledgement is needed.

- **Mode Source**

Indicates the alarm mode (Enable/Silence/Disable/Shelved) of the most urgent alarm that is in the InAlarm state or waiting to be Acked. If alarms are configured for an attribute, but no alarms are in the InAlarm state or waiting to be Acked, the value is the same

as the AlarmMode of the object.

- **Severity Source**

Indicates the severity as an integer (1-4) of the most urgent alarm current in an InAlarm state. If no alarms are in an InAlarm state or waiting to be acknowledged, the value is 0.

- **Shelved Source**

Indicates the current Shelved status (True/False) of the most urgent alarm that is in the InAlarm state or waiting to be Acked. If no alarms are in the InAlarm state or waiting to be Acked, the value is False.

To set Alarm Border animation for individual alarms, specify references to the following alarm attributes or tags:

- InAlarm attribute
- Acked attribute
- Mode attribute
- Severity attribute
- Shelved attribute

Alarm Border animation subscribes to these attributes or tags. Based on the alarm state of these attributes or tags, Alarm Border animation is applied to the graphic element in run time.

### To configure Alarm Border animation

1. Open a graphic in the Industrial Graphic Editor.
2. Select the graphic to show the graphic elements listed in the **Elements** pane of the Industrial Graphic Editor.
3. Select a graphic element from the **Elements** list to apply Alarm Border animation.
4. Click **Add Animation** to show the list of animation types.
5. Select **Alarm Border** from the list of **Visualization** animations.  
The **Alarm Border** dialog box appears with a set of configuration options.
6. Select either **Use Standard Alarm-Urgency References** or **Use Customized Alarm-Urgency References**.
  - **To use Standard Alarm-Urgency References**  
Click **Browse** and select an attribute, tag, or object name, then Click **OK**.  
Both direct and relative references to an object or tag are supported. An expression cannot be used to reference the object.
  - **To use Customized Alarm-Urgency References**  
Click **Browse** and select an attribute, a graphic element, or a tag name for all Source fields shown beneath **Use Customized Alarm Urgency References**, then click **OK**.  
All fields must contain values and cannot be left blank. Expressions, external references, and custom properties can be entered in all fields.
7. Enter a custom property, a constant (True/False), an external reference, or an expression in the **Show Alarm Indicator** field to set the condition when an alarm indicator icon is shown or hidden.

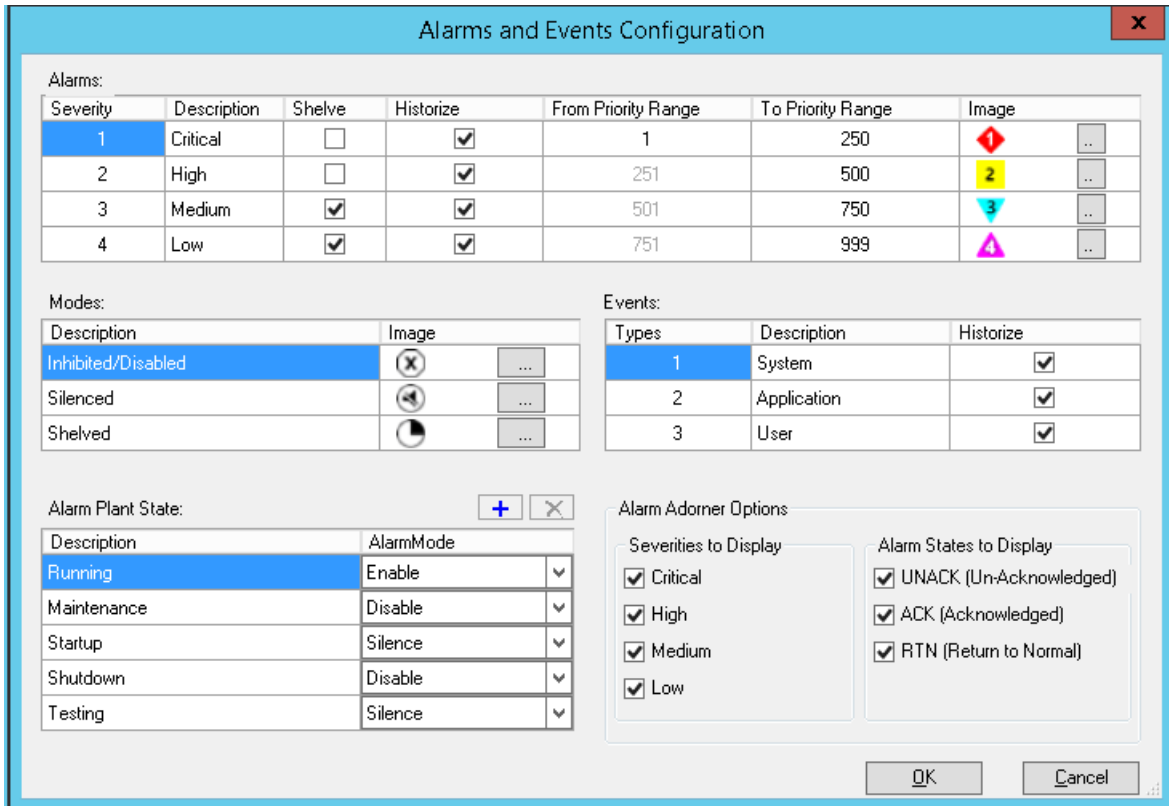
## Configuring Optional Alarm Border Animation Characteristics

You can complete a set of optional tasks to customize the appearance of Alarm Border animation.

## Changing Alarm Border Indicator Icons

Alarm border animation shows an indicator icon at the top left corner of the border with alarm severity as a number from 1 to 4 if your HMI/SCADA software supports it. Other indicator images represent alarm suppressed, silenced and shelved modes.

A default Alarm Border indicator image is assigned to each alarm mode and severity level. The default images appear in the **Image** fields of the **Alarm and Event Priority Mapping and Historization** dialog box. The images are saved in an XML file. For details about the location of the file, refer to your HMI/SCADA software documentation.



The default Alarm Border indicator images can be replaced by custom images. Supported image file types include .bmp, .gif, .jpg, .jpeg, .tif, .tiff, .png, .ico and .emf.

### To replace Alarm Border indicator images

1. Create Alarm Border indicator images that will replace the default indicator images.
2. On the **Galaxy** menu, click **Configure** and click **Alarm Priority Mapping**. The **Alarm and Event Priority Mapping and Historization** dialog box appears.
3. Click the **Search** button next to the Alarm Border indicator or mode image to be replaced. The **Open** dialog box appears to locate the replacement Alarm Border indicator images.
4. Go to the folder containing the replacement images.
5. Select an image file and click **Open**.
6. Verify the new Alarm Border indicator image replaced the original image in the **Alarm and Event Priority Mapping and Historization** dialog box.

At run time, Alarm Border animation reads the XML file from the location where it is stored. If images are not available from the XML file, an Alarm Indicator does not appear during Alarm Border animation.

## Modify Alarm Border Animation Element Styles

The color and fill pattern of alarm borders are set by the Outline properties of a set of AlarmBorder Element Styles. The following table shows the Element Styles applied to Alarm Border animations by alarm severity and alarm state.

The assignment of these Element Styles to alarm conditions cannot be changed. Only the assigned Element Style's Outline properties can be changed to modify the line pattern, line weight, and line color of alarm borders.

Alarm Severity	Alarm State	Element Style
1	UnAcknowledged	AlarmBorder_Critical_UNACK
1	Acknowledged	AlarmBorder_Critical_ACK
1	Return To Normal	AlarmBorder_Critical_RTN
2	UnAcknowledged	AlarmBorder_High_UNACK
2	Acknowledged	AlarmBorder_High_ACK
2	Return To Normal	AlarmBorder_High_RTN
3	UnAcknowledged	AlarmBorder_Medium_UNACK
3	Acknowledged	AlarmBorder_Medium_ACK
3	Return To Normal	AlarmBorder_Medium_RTN
4	UnAcknowledged	AlarmBorder_Low_UNACK
4	Acknowledged	AlarmBorder_Low_ACK
4	Return To Normal	AlarmBorder_Low_RTN
All	Suppressed	AlarmBorder_Suppressed
All	Shelved	AlarmBorder_Shelved
All	Silenced	AlarmBorder_Silenced

## Configuring a Percent Fill Horizontal Animation

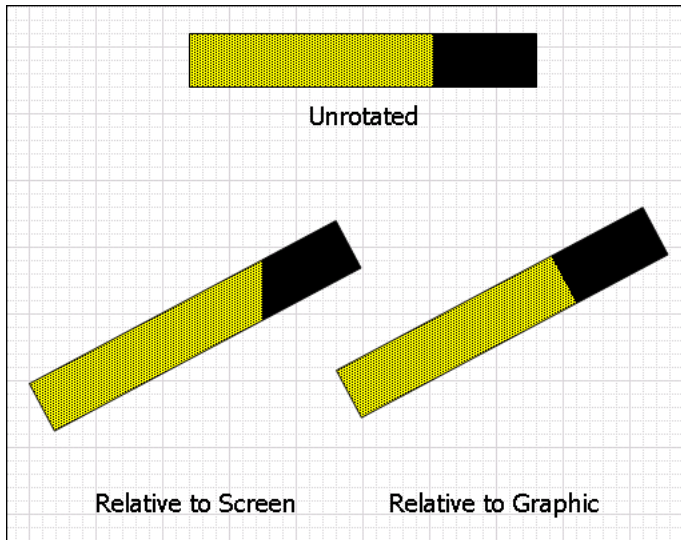
You can configure an element with a percent fill horizontal animation.

Besides specifying the expressions that determine how much of the element is filled at run time, you can also specify:

- **Fill direction:** from left to right or right to left.
- **Unfill color:** the style of the background when the element has 0 percent filling.



- **Fill orientation:** if the filling is in relation to the element or to the screen. This affects how the fill appears if the orientation of the element changes. If the fill is in relation to the screen and the element or graphic are rotated, the fill remains in relation to the screen.



**Note:** The fill orientation is a common setting to the percent fill horizontal and percent fill vertical animations.

You can also preview how the percent fill horizontal animation appears at run time.

**To configure an element with a percent fill horizontal animation**

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **% Fill Horizontal**. The percent fill horizontal animation is added to the Animation list and the **% Fill Horizontal** configuration panel appears.
4. Specify the settings. Do one or more of the following:
  - In the **Analog** box, type an analog value, attribute reference or expression.
  - In the **Value - At Min Fill** box, type an analog value, attribute reference or expression that causes the minimum percent of filling at run time.
  - In the **Value - At Max Fill** box, type an analog value, attribute reference or expression that causes the maximum percent of filling at run time.
  - In the **Fill - Min%** box, type an analog value, attribute reference or expression to specify the minimum percent of filling.
  - In the **Fill - Max%** box, type an analog value, attribute reference or expression to specify the maximum percent of filling.
  - In the **Colors** area, click the:
    - Fill Color** box to select a style from the **Select FillColor** dialog box. This is the fill style of the element.
    - Unfilled Color** box to select a style from the **Select FillColor** dialog box. This is the unfilled fill style of the element.
 For more information, see *Setting Style* on page 103.
5. In the **Direction** area, select:
  - **Right** - to fill from left to right.

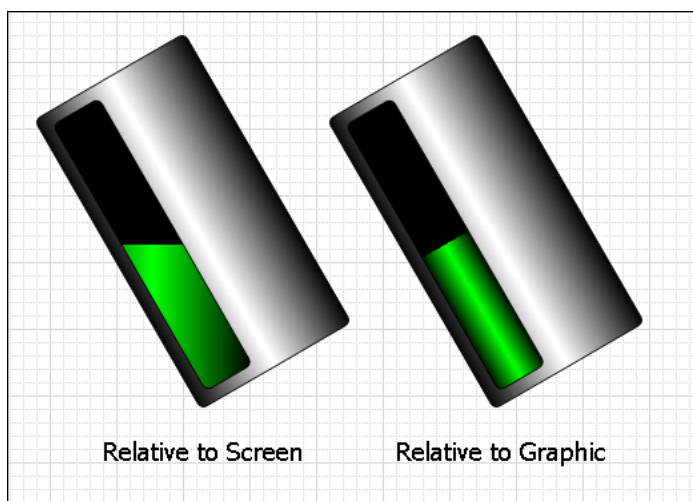
- **Left** - to fill from right to left.
6. In the **Orientation** area, select:
    - **Relative to Graphic** - so that the filling is in relation to the element and the filling rotates with the element.
    - **Relative to Screen** - so that the filling is in relation to the screen and the filling does not rotate with the element.
  7. You can preview your configuration by using the slider in the **Preview** area. Drag the slider to see how different values affect the appearance at run time.
  8. Click **OK**.

## Configuring a Percent Fill Vertical Animation

You can configure an element with a percent fill vertical animation.

Besides specifying the expressions that determine how much of the element is filled at run time, you can also specify:

- **Fill direction:** from lower to top or top to lower.
- **Unfill color:** the style of the background when the element has 0 percent filling.
- **Fill orientation:** if the filling is in relation to the element or to the screen. This affects how the fill appears if the orientation of the element changes. If the fill is in relation to the screen and the element or graphic are rotated, the fill remains in relation to the screen.



**Note:** The fill orientation is a common setting to the percent fill horizontal and percent fill vertical animations.

### To configure an element with a percent fill vertical animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **% Fill Vertical**. The percent fill vertical animation is added to the Animation list and the **% Fill Vertical** configuration panel appears.
4. In the **Analog** box, type an analog value, attribute reference or expression.
5. In the **Value-At Min Fill** box, type an analog value, attribute reference or expression that causes the minimum percent of filling at run time.

6. In the **Value-At Max Fill** box, type an analog value, attribute reference or expression that causes the maximum percent of filling at run time.
  7. In the **Fill-Min%** box, type an analog value, attribute reference or expression to specify the minimum percent of filling.
  8. In the **Fill-Max %** box, type an analog value, attribute reference or expression to specify the maximum percent of filling.
  9. In the **Colors** area, click the:
    - **Fill Color** box to select a style from the **Select FillColor** dialog box. This is the fill style of the element.
    - **Unfilled Color** box to select a style from the **Select FillColor** dialog box. This is the unfilled fill style of the element.
- For more information, see *Setting Style* on page 103.
10. In the **Direction** area, select:
    - **Up** - to fill from lower to top.
    - **Down** - to fill from top to lower.
  11. In the **Orientation** area, select:
    - **Relative to Graphic** - so that the filling is in relation to the element and the filling rotates with the element.
    - **Relative to Screen** - so that the filling is in relation to the screen and the filling does not rotate with the element.
  12. You can preview your configuration by using the slider in the **Preview** area. Drag the slider to see how different values affect the appearance at run time.
  13. Click **OK**.

## Configuring a Horizontal Location Animation

You can configure an element with a horizontal location animation.

### To configure an element with a horizontal location animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Location Horizontal**. The horizontal location animation is added to the Animation list and the **Location Horizontal** configuration panel appears.
4. In the **Analog** box, type an analog value, attribute reference or expression.
5. In the **Value-At Left End** box, type an analog value, attribute reference or expression that corresponds to the offset specified by the **Movement-To Left** value.
6. In the **Value-At Right End** box, type an analog value, attribute reference or expression that corresponds to the offset specified by the **Movement-To Right** value.
7. In the **Movement-To Left** box, type an analog value, attribute reference or expression for the maximum offset to the left.
8. In the **Movement-To Right** box, type an analog value, attribute reference or expression for the maximum offset to the right.
9. Click **OK**.

## Configuring a Vertical Location Animation

You can configure an element with a vertical location animation.

### To configure an element with a vertical location animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Location Vertical**. The vertical location animation is added to the Animation list and the **Location Vertical** configuration panel appears.
4. In the **Analog** box, type an analog value, attribute reference or expression.
5. In the **Value - At Top** box, type an analog value, attribute reference or expression that corresponds to the offset specified by the **Movement - Up** value.
6. In the **Value - At lower** box, type an analog value, attribute reference or expression that corresponds to the offset specified by the **Movement - Down** value.
7. In the **Movement - Up** box, type an analog value, attribute reference or expression for the maximum offset upwards.
8. In the **Movement - Down** box, type an analog value, attribute reference or expression for the maximum offset downwards.
9. Click **OK**.

## Configuring a Width Animation

You can configure an element with a width animation. You can also specify if the element is to be anchored to its left, center, right side or origin.

### To configure an element with a width animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Width**. The width animation is added to the Animation list and the **Width** configuration panel appears.
4. In the **Analog** box, type an analog value, attribute reference or expression.
5. In the **Value-At Min Size** box, type an analog value, attribute reference or expression that corresponds to the minimum width specified by the **Width-Min%** value.
6. In the **Value-At Max Size** box, type an analog value, attribute reference or expression that corresponds to the maximum width specified by the **Width-Max%** value.
7. In the **Width-Min%** box, type an analog value, attribute reference or expression for the minimum width in percent of the original element.
8. In the **Width-Max%** box, type an analog value, attribute reference or expression for the maximum width in percent of the original element.
9. In the **Anchor** area, select:
  - **Left** - to specify that the left of the element is anchored.
  - **Center** - to specify that the horizontal center of the element is anchored.
  - **Right** - to specify that the right side of the element is anchored.
  - **Origin** - to specify that the origin of the element is anchored.
10. Click **OK**.

## Configuring a Height Animation

You can configure an element with a height animation. You can also specify if the element is to be anchored to its top side, middle, lower side or origin.

### To configure an element with a height animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Height**. The height animation is added to the Animation list and the **Height** configuration panel appears.
4. In the **Analog** box, type an analog value, attribute reference or expression.
5. In the **Value-At Min Size** box, type an analog value, attribute reference or expression that corresponds to the minimum height specified by the **Height-Min%** value.
6. In the **Value-At Max Size** box, type an analog value, attribute reference or expression that corresponds to the maximum height specified by the **Height-Max%** value.
7. In the **Height-Min%** box, type an analog value, attribute reference or expression for the minimum height in percent of the original element.
8. In the **Height-Max%** box, type an analog value, attribute reference or expression for the maximum height in percent of the original element.
9. In the **Anchor** area, select:
  - **Top** - to specify that the top side of the element is anchored.
  - **Middle** - to specify that the vertical center of the element is anchored.
  - **Lower** - to specify that the lower side of the element is anchored.
  - **Origin** - to specify that the origin of the element is anchored.
10. Click **OK**.

## Configuring a Point Animation

Point animation changes the X and Y coordinate values of one or more selected points on a graphic or graphic element. During run time, the X and the Y coordinates of a selected point are set to an expression or reference that evaluates to a calculated real floating point value.

Point animation supports negative floating point values, which may cause the animation to go out of scope of the visualization window. In the case when a point's expression evaluates to a null value or causes an exception, animation stops and the point retains its original value.

The X and Y coordinates of a point can be configured as a pair or individually. If only the X coordinate of a point is configured, then the Y coordinate value is kept constant. The animation shows the point of the graphic element traversing the X axis. Likewise, if only the Y coordinate of a point is configured, then the animation shows the point traversing the Y axis with the point's X axis value held constant.

After selecting point animation, a list of configurable points is retrieved from the graphic element based on the following conditions.

- If the graphic element is a multi-point graphic type (Line, HV/Line, Polyline, Curve, Polygon, Closed curve), animation control points appear on the graphic element in preview mode.
- If the graphic element is not a supported multi-point graphic, then the top left X and Y coordinate of the graphic element is selected as the animation point.

In the case of an element group consisting of several graphic elements, the animation point is the top left corner of the rectangle around all grouped elements.

### To configure point animation

1. Open the graphic in Industrial Graphic Editor.
2. Select a graphic element.
3. On the **Special** menu, click **Edit Animations**.

The **Edit Animations** dialog box appears.

4. Click the **Add Animation** button to show a list of Visualization and Interaction animations.
5. Select **Point** from the Visualization animation list.

The **Point** dialog box appears with a list of points and a preview of the points on the graphic. The list shows each point as a pair of X and Y fields to enter an expression or a reference that evaluates to a floating point value.

6. Select a point from the list of points.

The selected point changes to orange in the preview of the graphic.

7. Enter an expression, constant, or reference in the Point field.
8. Repeat steps 6-7 to animate other points in the graphic.
9. Save your changes.

## Configuring an Orientation Animation

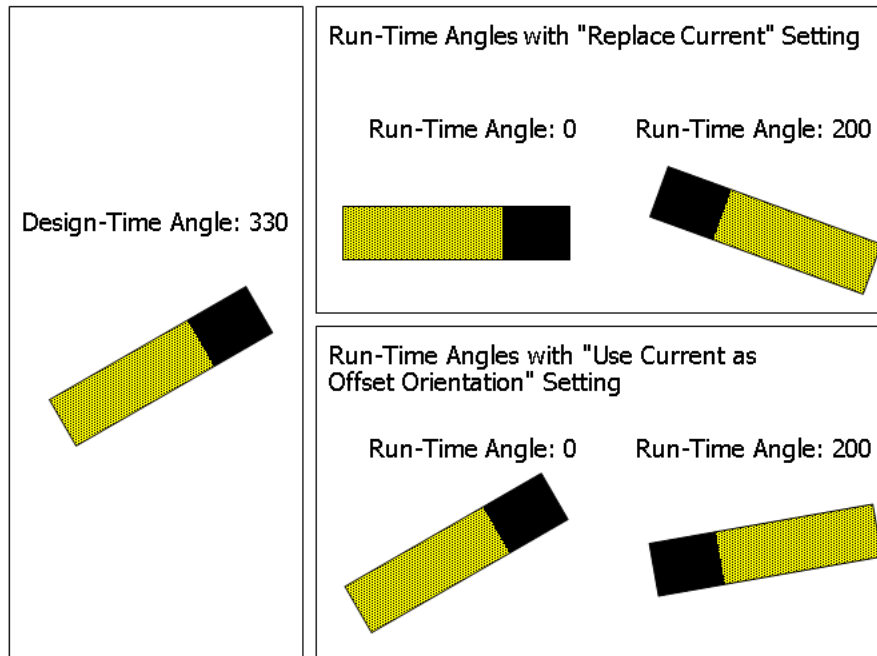
You can configure an element with an orientation animation. You can also:

- Specify a different orientation origin.
- Ignore or accept the design-time orientation of the element on the canvas.
- Preview the orientation at run time with a slider.

### To configure an element with an orientation animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Orientation**. The orientation animation is added to the Animation list and the **Orientation** configuration panel appears.
4. In the **Analog** box, type an analog value, attribute reference or expression.
5. In the **Value-At CCW End** box, type an analog value, attribute reference or expression that corresponds to the maximum angle in degrees for the counter-clockwise orientation as specified by the **Orientation-CCW** value.
6. In the **Value-At CW End** box, type an analog value, attribute reference or expression that corresponds to the maximum angle in degrees for the counter-clockwise orientation as specified by the **Orientation-CW** value.
7. In the **Orientation-CCW** box, type an analog value, attribute reference or expression for the maximum orientation in counter-clockwise direction in degrees.
8. In the **Orientation-CW** box, type an analog value, attribute reference or expression for the maximum orientation in clockwise direction in degrees.
9. In the **Orientation Offset** area, select:
  - **Replace Current** to ignore the design-time orientation of the element as it appears on the canvas and to use absolute orientation.

- **Use Current as Offset Orientation** to orientate the element at run time in relation to its design-time orientation on the canvas.



10. If you use current as offset orientation, you can type an offset value in the text box next to **Use Current as Offset Orientation**. This affects the orientation of the element on the canvas.
11. In the **Current Relative Origin** area, type values in the **dX** and **dY** boxes to specify the rotation origin as offset from the element center point. This affects the point of origin of the element on the canvas.
12. You can preview the orientation and how run-time values affect the appearance of the element, by dragging the slider in the **Preview** area.
13. Click **OK**.

## Configuring a Value Display Animation

You can configure an element with a value display animation. You can show:

- A Boolean value as a Message.
- An Analog value.
- A string value.
- A time or date value.
- The tag name, hierarchical name or contained name of the hosting object.

## Configuring a Boolean Value Display Animation

You can configure an element to show a Boolean value as a message.

### To configure an element with a Boolean value display animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.

3. Click the **Add** icon and select **Value Display**. The value display animation is added to the Animation list and the **Value Display** state selection panel appears.
4. Click the **Boolean** button. The **Boolean Value Display** configuration panel appears.
5. In the **Boolean** box, type a Boolean value, attribute reference or expression.
6. In the **True Message** box, type a value, attribute reference or expression for the text display when the expression is true.
7. In the **False Message** box, type a value, attribute reference or expression for the text display when the expression is false.
8. Click **OK**.

## Configuring an Analog Value Display Animation

You can configure a text element, TextBox, or button to show an analog value. You can also configure an analog value display animation for a grouped element when the TreatAsIcon property is True.

### To configure an element with an analog value display animation

1. Select the text element, TextBox, Button, or grouped element that you want to configure Analog Value Display animation.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Value Display**. The value display animation is added to the Animation list and the **Value Display** state selection panel appears.
4. Click the **Analog** button. The **Analog Value Display** configuration panel appears.
5. In the **Analog** box, type an analog value, attribute reference or expression.
6. Click the **Text Format** drop-down list and select a global number style.
  - **Format String** is the default numeric format, which includes a text field to enter characters that specify a number format. For more information about assigning a number format using typed characters, see *Enter Input Numbers in U.S. Format* on page 127.
  - Selecting custom from the **Text Format** list shows another drop-down list to select a number format.
    - **Fixed Width**: Appears on the **Analog Value Display** dialog box for every Custom number format. When selected, the run-time number value will not exceed the text length specified for Value Display animation.
    - **Precision**: Appears on the **Analog Value Display** dialog box for the **Fixed Decimal** and **Exponential Custom** number formats. **Precision** sets the precision of the fractional part of a number to the right of the decimal point.
    - **Bits From** and **To**: Appear on the **Analog Value Display** dialog box for the **Hex** and **Binary** number formats. **Bits From** sets the starting bit position of a hex or binary number shown during run time. **To** sets the ending bit position of a hex or binary number shown during run time.
7. Click **OK**.

Except for the **Format String** and **Custom** text styles, all other text styles are global number styles that do not need further configuration.

For more information about the listed global number formats, see *Setting Global Number Styles*.



## Configuring Value Display Animation with the FormatStyle Property

When Value Display or User Input animation have a text format configured with a number style from your HMI/SCADA software, numeric data shown during run time is formatted in accordance to the number style selected for the animation.

Changes to a number style are not propagated during run time. Any changes to global number styles become effective only after the HMI/SCADA software is restarted.

A number style can be changed during run time using the FormatStyle property. FormatStyle is a text element run-time property that displays the name of the current applied global number style. The value of the FormatStyle property can be set as the active number style of Value Display and User Input animation during run time .

- If the name applied to the FormatStyle property during run time is the name of a global number style, the text element is reformatted using the new applied number style for Value Display or User Input animation.
- If the FormatStyle property is assigned a value that does not match any global number style, the value of FormatStyle remains unchanged and a warning message is logged to SMC logger.
- If FormatStyle is set to an empty string, the text format of User Input and Value Display animation reverts to the value specified for **Text Format** during design time.
- If a text element is inside a group in which the **TreatAsIcon** property is set to True, then the group's text format overrides the first text child element for Value Display or User Input animation.
- If a text element's FormatStyle property changes in run time, the new number style is used to format the text element. Because a graphic group does not have TextFormat and FormatStyle properties, using the FormatStyle property is the only way to change the format of text in run time.

## Configuring a String Value Display Animation

You can configure an element to show a string value.

### To configure an element with a string value display animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Value Display**. The value display animation is added to the Animation list and the **Value Display** state selection panel appears.
4. Click the **String** button. The **String Value Display** configuration panel appears.
5. In the **String** box, type a string value, attribute reference or expression.
6. Click **OK**.

## Configuring a Time Value Display Animation

You can configure an element to show a time value.

Use the following letters to set the time format:

h	The one or two-digit hour in 12-hour format.
hh	The two-digit hour in 12-hour format. Single digit values are preceded by a zero.
H	The one or two-digit hour in 24-hour format.
HH	The two-digit hour in 24-hour format. Single digit values are preceded by a zero.

t	The one-letter AM/PM abbreviation ("AM" appears as "A").
tt	The two-letter AM/PM abbreviation ("AM" appears as "AM").
m	The one or two-digit minute.
mm	The two-digit minute. Single digit values are preceded by a zero.
s	The one or two-digit seconds.
ss	The two-digit seconds. Single digit values are preceded by a zero.
d	The one or two-digit day.
dd	The two-digit day. Single digit day values are preceded by a zero.
ddd	The three-character day-of-week abbreviation.
dddd	The full day-of-week name.
M	The one or two-digit month number.
MM	The two-digit month number. Single digit values are preceded by a zero.
MMM	The three-character month abbreviation.
MMMM	The full month name.
y	The one-digit year (2001 appears as "1").
yy	The last two digits of the year (2001 appears as "01").
yyyy	The full year (2001 appears as "2001").

The format for elapsed time is:

```
[-] [DDDDDD] [HH:MM:]SS [.ffffff]
```

Use the following letters to set the elapsed time format:

DDDDDD	The number of days. Valid values are 0 to 999999.
HH	The two-digit hour in 24-hour format. Single digit values are preceded by a zero. Valid values are 00 to 23.
MM	The two-digit month number. Single digit values are preceded by a zero. Valid values are 00 to 59.
SS	The two-digit seconds. Single digit values are preceded by a zero. Valid values are 00 to 59.
ffffff	Optional fractional seconds to right of decimal, and can be one through seven digits.

---

**Note:** You can use any other characters, except "g" in the property. These characters then appear at design time and run time in the control.

---

### To configure an element with a time value display animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.

3. Click the **Add** icon and select **Value Display**. The value display animation is added to the Animation list and the **Value Display** state selection panel appears.
4. Click the **Time** button. The **Time Value Display** configuration panel appears.
5. In the **Time or Elapsed Time** box, type a time or elapsed time value, attribute reference or expression.
6. In the **Text Format** box, type a format for the value output. If you change this value, the TextFormat property of the element also changes.
7. Click **OK**.

## Configuring a Name Display Animation

You can configure an element to show the tag name, hierarchical name or contained name of the AutomationObject that is hosting it.

For example if the AutomationObject hosting the graphic is named Valve\_001 and Valve\_001 is contained in Pump\_001 and has a contained name of InletValve, then configuring an element with the value display animation with:

- **Tag Name** shows Valve\_001 at run time
- **Hierarchical Name** shows Pump\_001.InletValve at run time
- **Contained Name** shows InletValve at run time

### To configure an element with a name display animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Value Display**. The value display animation is added to the Animation list and the **Value Display** state selection panel appears.
4. Click the **Name** button. The **Name Display** configuration panel appears.
5. Select:
  - **Tag Name** to show the tag name of the hosting AutomationObject.
  - **Hierarchical Name** to show the hierarchical name of the hosting AutomationObject.
  - **Contained Name** to show the contained name of the hosting AutomationObject.
6. Click **OK**.

## Configuring a Tooltip Animation

You can configure an element with a tooltip animation.

### To configure an element with a tooltip animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Tooltip**. The tooltip animation is added to the Animation list and the **Tooltip** configuration panel appears.
4. In the **Expression** box, type:
  - A static value and make sure the **Input Mode** icon is set to static.
  - An attribute reference or expression and make sure the **Input Mode** icon is set to attribute or reference.

5. Click **OK**.

## Configuring a Disable Animation

You can configure an element with a disable animation. This lets you disable user interaction with an element depending on a run-time value or expression.

### To configure an element with a disable animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Disable**. The disable animation is added to the Animation list and the **Disable** configuration panel appears.
4. In the **Boolean** box, type a Boolean numeric value, attribute reference or expression.
5. In the **Disabled When Expression is** area, select:
  - **True, 1, On** in which case the element is disabled at run time whenever the expression is true, and enabled whenever the expression is false.
  - **False, 0, Off** in which case the element is disabled at run time whenever the expression is false, and enabled whenever the expression is true.
6. Click **OK**.

## Configuring a User Input Animation

You can configure an element with a user input animation for the following data types:

- Boolean
- Analog (integer, float, double)
- String
- Time
- Elapsed time

## Configuring a User Input Animation for a Discrete Value

You can configure an element with a user input animation for a Boolean value.

### To configure an element with a user input animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **User Input**. The user input animation is added to the Animation list and the **User Input** state selection panel appears.
4. Click the **Boolean** button. The **Boolean Value User Input** configuration panel appears.
5. Specify the options. Do one or more of the following:
  - In the **Boolean** box, type an attribute reference or browse for one by using the browse button.
  - In the **Message to User** box, type a value, attribute reference or expression. This is the text that appears as prompt on the Boolean value input dialog box at run time.
  - In the **Prompt - True Message** box, type a value, attribute reference or expression. This is the text that appears on the button that causes the attribute to be set to true.

- In the **Prompt - False Message** box, type a value, attribute reference or expression. This is the text that appears on the button that causes the attribute to be set to false.
  - Specify that the input dialog box appears by pressing a key or key combination. In the **Shortcut** area. Select a shortcut key in the **Key** list. Select **CTRL** and/or **SHIFT** to combine the shortcut key with the CTRL key and/or SHIFT key.
  - If you don't want the discrete value display element to show the True Message and False Message, select **Input Only**.
  - In the **Display Value - True Message** box, type a value, attribute reference or expression. This is the text that appears on the canvas when the associated attribute is true.
  - In the **Display Value - False Message** box, type a value, attribute reference or expression. This is the text that appears on the canvas when the associated attribute is false.
  - Make sure that the input modes of all boxes are set correctly. Click the **Input Mode** icon to set a static value or an attribute reference or expression.
6. Click **OK**.

## Configuring a User Input Animation for an Analog Value

You can configure an element with a user input animation for an analog value.

### To configure an element with a user input animation for an analog value

1. Select the text element, TextBox, Button, or grouped element that you want to configure for User Input animation.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon from the **Animations** pane and select **User Input**.  
User Input is added to the **Interaction** list and **User Input** state selection panel appears.
4. Click the **Analog** button. The **Analog Value User Input** configuration panel appears.
5. In the **Analog** box, type an attribute reference or browse for one by using the browse button.
6. In the **Message to User** box, type a value, attribute reference, or expression. This text appears to prompt for the analog value input dialog box at run time.
7. Make sure that the input mode of the **Message to User** box is set correctly. Click the **Input Mode** icon to set a static value or an attribute reference or expression.
8. If you want to restrict the range of input values, you can do so in the **Value Limits** area by:
  - First selecting **Restrict Values**.
  - Enter values, attribute references, or expressions in the **Minimum** and **Maximum** boxes.
9. In the **Shortcut** area, specify that an **Input** dialog box appears by pressing a key or key combination. Select a shortcut key from the **Key** list. Select **CTRL** and/or **SHIFT** to combine the shortcut key with the Ctrl key and/or Shift key.
10. Select **Input Only** to restrict the text of a graphic from showing the current value of the reference attribute.  
If unchecked, the text of a graphic shows the current value of the reference attribute.
11. Select **Use Keypad** to show a keypad during run time for the user to type a data value.
12. Click the Text Format drop-down list and select a global number format.

For more information about the listed global number styles, see Setting Global Number Styles.

**Format String** is the default numeric format, which includes a text entry field to assign a number format using four characters:

<b>Numeric Format Character</b>	<b>Description</b>
Zero, (0)	Represents a digit at each specified position of a real number. Forces leading zeros to the integer part of a number and trailing zeros to the fractional part of a number.
Pound sign, (#)	Represents a digit at that position within a number.
Comma, (,)	Inserts a comma at the specified position of a real number.
Decimal point, (.)	Inserts a decimal point at the specified position of a real number.

Except for the Format String and Custom text styles, all other text styles are global number styles that do not need further configuration.

13. Click **OK**.

## Configuring a User Input Animation for a String Value

You can configure an element with a user input animation for a string value.

### To configure an element with a user input animation for a string value

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **User Input**. The user input animation is added to the Animation list and the **User Input** state selection panel appears.
4. Click the **String** button. The **String Value User Input** configuration panel appears.
5. In the **String** box, type an attribute reference or browse for one by using the browse button.
6. In the **Message to User** box, type a value, attribute reference or expression. This is the text that appears as prompt on the string value input dialog box at run time.
7. Make sure that the input mode of the **Message to User** box is set correctly. Click the **Input Mode** icon to set a static value or an attribute reference or expression.
8. You can specify that the **Input** dialog box appears by pressing a key or key combination. In the **Shortcut** area. Select a shortcut key in the **Key** list. Select **CTRL** and/or **SHIFT** to combine the shortcut key with the CTRL key and/or SHIFT key.
9. If you don't want the string value display element to show the string input result on the canvas, select **Input Only**.
10. If you want to use the keypad to type the string value, select **Use Keypad**.
11. If you select **Input Only** and want to see placeholders during the input at run time, select **Echo Characters**.
12. If you are configuring a password input:
  - Select **Password**.
  - Type in the replacement character in the adjacent box.
  - Select **Encrypt** if you want to encrypt the string that holds the password.

---

**Important:** Password encryption only works within the context of HMI/SCADA software applications that support password encryption. Do not encrypt the string if you want to pass it to an external security system, such as the operating system or a SQL Server database. The external security system cannot read the encrypted password string and access will fail.

---

13. Click **OK**.

## Configuring a User Input Animation for a Time Value

You can configure an element with a user input animation for a time value.

### To configure an element with a user input animation for a time value

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **User Input**. The user input animation is added to the Animation list and the **User Input** state selection panel appears.
4. Click the **Time** button. The **Time Value User Input** configuration panel appears.
5. In the **Time** box, type an attribute reference or browse for one by using the browse button.
6. In the **Message to User** box, type a value, attribute reference or expression. This is the text that appears as prompt on the time value input dialog box at run time.
7. Make sure that the input mode of the **Message to User** box is set correctly. Click the **Input Mode** icon to set a static value or an attribute reference or expression.
8. Specify that the **Input** dialog box appears by pressing a key or key combination. In the **Shortcut** area. Select a shortcut key in the **Key** list. Select **CTRL** and/or **SHIFT** to combine the shortcut key with the Ctrl key and/or Shift key.
9. If you don't want the time value display element to show the time input result on the canvas, select **Input Only**.
10. To use the current date and time as default, select **Use Current Date/Time as Default**.
11. Select:
  - **Use Input Dialog** to use the **Time User Input** dialog box at run time to type date and time values in individual boxes.
  - **Use Calendar** to use the **Time User Input** dialog box at run time to type a date with the calendar control.
12. If you select **Use Input Dialog** to type the time value, you can select:
  - **Date and Time** to type date and time.
  - **Date** to only type a date.
  - **Time** to only type a time.Select **Show Seconds** if you also want to input seconds.
13. If you want to format your text after input, type a valid text format in the **Text Format** box. Use the following letters to set the time format:

h	The one or two-digit hour in 12-hour format.
hh	The two-digit hour in 12-hour format. Single digit values are preceded by a zero.
H	The one or two-digit hour in 24-hour format.
HH	The two-digit hour in 24-hour format. Single digit values are preceded by a zero.
t	The one-letter AM/PM abbreviation ("AM" appears as "A").
tt	The two-letter AM/PM abbreviation ("AM" appears as "AM").

m	The one or two-digit minute.
mm	The two-digit minute. Single digit values are preceded by a zero.
s	The one or two-digit seconds.
ss	The two-digit seconds. Single digit values are preceded by a zero.
d	The one or two-digit day.
dd	The two-digit day. Single digit day values are preceded by a zero.
ddd	The three-character day-of-week abbreviation.
dddd	The full day-of-week name.
M	The one or two-digit month number.
MM	The two-digit month number. Single digit values are preceded by a zero.
MMM	The three-character month abbreviation.
MMMM	The full month name.
y	The one-digit year (2001 appears as "1").
yy	The last two digits of the year (2001 appears as "01").
yyyy	The full year (2001 appears as "2001").

---

**Note:** You can use any other characters, except "g" in the property. These characters then appear at design time and run time in the control.

---

14. Click **OK**.

## Configuring a User Input Animation for an Elapsed Time Value

You can configure an element with a user input animation for an elapsed time value.

### To configure an element with a user input animation for an elapsed time value

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **User Input**. The user input animation is added to the Animation list and the **User Input** state selection panel appears.
4. Click the **Elapsed Time** button. The **Elapsed Time Value User Input** configuration panel appears.
5. In the **Elapsed Time** box, type an attribute reference or browse for one by using the browse button.
6. In the **Message to User** box, type a value, attribute reference or expression. This is the text that appears as prompt on the elapsed time value input dialog box at run time.
7. Make sure that the input mode of the **Message to User** box is set correctly. Click the **Input Mode** icon to set a static value or an attribute reference or expression.
8. Specify that the **Input** dialog box appears by pressing a key or key combination. In the **Shortcut** area. Select a shortcut key in the **Key** list. Select **CTRL** and/or **SHIFT** to combine the shortcut key with the CTRL key and/or SHIFT key.
9. If you don't want the elapsed time value display element to show the time elapsed input result on the canvas, select **Input Only**.



10. Select **Use Dialog** to use the **Elapsed Time User Input** dialog box to type the elapsed time value at run time.
11. If you select **Use Dialog** to type the elapsed time value, you can optionally select:
  - Show Days** if you also want to input days.
  - Show Milliseconds** if you also want to input milliseconds.
12. Click **OK**.

## Configuring a Horizontal Slider Animation

You can configure an element with a horizontal slider animation. This lets you drag an element at run time in horizontal direction and write a corresponding value back to an attribute.

### To configure an element with a horizontal slider animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Slider Horizontal**. The horizontal slider animation is added to the Animation list and the **Slider Horizontal** configuration panel appears.
4. In the **Analog** box, type an attribute reference or browse for one by using the browse button.
5. In the **Value - Left Position** box, type an analog value, attribute reference or expression that corresponds to the offset specified by the **Movement - To Left** value.
6. In the **Value - Right Position** box, type an analog value, attribute reference or expression that corresponds to the offset specified by the **Movement - To Right** value.
7. In the **Movement - To Left** box, type an analog value, attribute reference or expression for the maximum offset to the left in pixels.
8. In the **Movement - To Right** box, type an analog value, attribute reference or expression for the maximum offset to the right in pixels.
9. You can select where the cursor is anchored to the element when it is dragged at run time. In the **Cursor Anchor** area, select:
  - **Left** to anchor the element at its left.
  - **Center** to anchor the element at its center point.
  - **Right** to anchor the element at its right side.
  - **Origin** to anchor the element at its point of origin.
10. You can select if position data from the slider is written continuously to the attribute, or only one time when the mouse button is released. In the **Write Data** area, select **Continuously** or **On mouse release**.
11. If you want a tooltip to appear on the element showing the current value during dragging, select **Show Tooltip**.
12. Preview the movement as it appears in run time by dragging the slider in the **Preview** area.
13. Click **OK**.

## Configuring a Vertical Slider Animation

You can configure an element with a vertical slider animation.

### To configure an element with a vertical slider animation

1. Select the element.

2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Slider Vertical**. The vertical slider animation is added to the Animation list and the **Slider Vertical** configuration panel appears.
4. In the **Analog** box, type an attribute reference or browse for one by using the browse button.
5. In the **Value - Top Position** box, type an analog value, attribute reference or expression that corresponds to the offset specified by the **Movement - Up** value.
6. In the **Value - lower Position** box, type an analog value, attribute reference or expression that corresponds to the offset specified by the **Movement - Down** value.
7. In the **Movement - Up** box, type an analog value, attribute reference or expression for the maximum offset upwards in pixels.
8. In the **Movement - Down** box, type an analog value, attribute reference or expression for the maximum offset downwards in pixels.
9. You can select where the cursor is anchored to the element when it is dragged at run time. In the **Cursor Anchor** area, select:
  - **Top** to anchor the element at its top side.
  - **Middle** to anchor the element at its middle point.
  - **Lower** to anchor the element at its lower side.
  - **Origin** to anchor the element at its point of origin.
10. You can select if position data from the slider is written continuously to the attribute, or only one time when the mouse button is released. In the **Write Data** area, select **Continuously** or **On mouse release**.
11. If you want a tooltip to appear on the element showing the current value during dragging, select **Show Tooltip**.
12. Preview the movement as it appears in run time by dragging the slider in the **Preview** area.
13. Click **OK**.

## Configuring a Pushbutton Animation

You can configure an element with a pushbutton animation to change Boolean, analog or string references.

### Configuring a Pushbutton Animation for a Boolean Value

You can configure an element with a pushbutton to change a Boolean value.

#### To configure an element with a pushbutton animation to change a Boolean value

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Pushbutton**. The pushbutton animation is added to the Animation list and the **Pushbutton** state selection panel appears.
4. Click the **Boolean** button. The **Boolean Pushbutton** configuration panel appears.
5. In the **Boolean** box, type a Boolean attribute reference or browse for one by using the browse button.
6. In the **Action** list, select:
  - **Direct** so the value becomes true when the element is clicked and the mouse button held. The value returns to false when the mouse button is released.

- **Reverse** so the value becomes false when the element is clicked and the mouse button held. The value returns to true when the mouse button is released.
  - **Toggle** so the value becomes true if it is false and false if it is true when the element is clicked.
  - **Set** so the value is set to true when the element is clicked.
  - **Reset** so the value is set to false when the element is clicked.
7. If you select **Toggle**, **Set** or **Reset** as action, you can configure the action to be performed when the mouse button is released instead of pressed down. To do this, select **On button release**.
  8. If you select Direct, Reverse, Reset or Set as action, you can configure the value to be written:
    - Continuously by selecting **Continuously while button is pressed**. Also specify the frequency the value is to be sent, by typing a value in the **Delay between value send** box.
    - One time by clearing **Continuously while button is pressed**.
  9. Specify that the pushbutton action is executed by pressing a key or key combination. In the **Shortcut** area. Select a shortcut key in the **Key** list. Select **CTRL** and/or **SHIFT** to combine the shortcut key with the Ctrl key and/or Shift key.
  10. Preview the pushbutton run-time behavior by clicking **Button** in the **Preview** area.
  11. Click **OK**.

## Configuring a PushButton Animation for an Analog Value

You can configure an element with a pushbutton to set an analog value.

### To configure an element with a pushbutton animation to set an analog value

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Pushbutton**. The pushbutton animation is added to the Animation list and the **Pushbutton** state selection panel appears.
4. Click the **Analog** button. The **Analog Pushbutton** configuration panel appears.
5. In the **Analog** box, type an attribute reference or browse for one by using the browse button.
6. From the **Action** list, select:
  - **Direct** so the value becomes Value1 when the element is clicked and the mouse button held. The value returns to Value2 when the mouse button is released.
  - **Toggle** so the value becomes Value1 if it is Value2 and Value2 if it is Value1 when the element is clicked.
  - **Set** so the value is set to Value1 when the element is clicked
  - **Increment** so the value is increased by Value1.
  - **Decrement** so the value is decreased by Value1.
  - **Multiply** so the value is multiplied with Value1.
  - **Divide** so the value is divided by Value1.
7. In the boxes **Value1** and, if applicable, **Value2**, type analog values, attribute references or references.
8. You can configure the value to be written when the mouse button is released instead. Select **On button release**. This does not apply if you select Direct as action.
9. You can configure the value to be written:

- Continuously by selecting the **Continuously while button is pressed**. Also specify the frequency the value is to be sent, by typing a value in the **Delay between value send** box.
- One time by clearing the **Continuously while button is pressed**.

This does not apply if you select Toggle as action.

10. Specify that the pushbutton action is executed by pressing a key or key combination. In the **Shortcut** area. Select a shortcut key in the **Key** list. Select **CTRL** and/or **SHIFT** to combine the shortcut key with the CTRL key and/or SHIFT key.
11. Preview the pushbutton run-time behavior by clicking **Button** in the **Preview** area. Click the button multiple times to preview the value changes over a period of time.
12. Click **OK**.

## Configuring a PushButton Animation for a String Value

You can configure an element with a pushbutton to set a string value.

### To configure an element with a pushbutton animation to set a string value

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Pushbutton**. The pushbutton animation is added to the Animation list and the **Pushbutton** state selection panel appears.
4. Click the **Analog** button. The **String Pushbutton** configuration panel appears.
5. In the **String** box, type an attribute reference or browse for one by using the browse button.
6. From the **Action** list, select:
  - **Direct** so the value becomes Value1 when the element is clicked and the mouse button held. The value returns to Value2 when the mouse button is released.
  - **Toggle** so the value becomes Value1 if it is Value2 and Value2 if it is Value1 when the element is clicked
  - **Set** so the value is set to Value1 when the element is clicked.
7. In the boxes **Value1** and, if applicable, **Value2**, type string values, attribute references or references.
8. Make sure that the input modes of the **Value1** and **Value2** boxes are set correctly. Click the **Input mode** icons to set a static values or an attribute references or expressions.
9. You can configure the value to be written when the mouse button is released instead. Select **On button release**. This does not apply if you select Direct as action.
10. You can configure the value to be written:
  - Continuously by selecting the **Continuously while button is pressed**. Also specify the frequency the value is to be sent, by typing a value in the **Delay between value send** box.
  - One time by clearing the **Continuously while button is pressed**.

This does not apply if you select Toggle as action.
11. Specify that the pushbutton action is executed by pressing a key or key combination. In the **Shortcut** area. Select a shortcut key in the **Key** list. Select **CTRL** and/or **SHIFT** to combine the shortcut key with the Ctrl key and/or Shift key.
12. Preview the pushbutton run-time behavior by clicking **Button** in the **Preview** area.
13. Click **OK**.

## Configuring an Action Script Animation

You can configure an element with an action script animation.

You can assign multiple action scripts to one element that are activated in different ways such as:

Use this...	To activate the action script when the...
<b>On Left Click/Key/Touch Down</b>	left mouse button or a specific key is pressed or a tap on a touch display.
<b>While Left Click/Key/Touch Down</b>	left mouse button or a specific key is pressed and held or the touch display is tapped and held.
<b>On Left/Key/Touch Up</b>	left mouse button or a specific key is released or the tap is released from a touch display.
<b>On Left Double Click/Double Tap</b>	left mouse button is double-clicked or the touch display is tapped twice in quick succession.
<b>On Right Click/Long Press</b>	right mouse button is pressed or a long press on the touch display.
<b>While Right Down</b>	right mouse button is pressed and held.
<b>On Right Up</b>	right button is released.
<b>On Right Double Click</b>	right mouse button is double-clicked.
<b>On Center Click</b>	center mouse button is pressed.
<b>While Center Click</b>	center mouse button is pressed and held.
<b>On Center Up</b>	center mouse button is released.
<b>On Center Double Click</b>	center mouse button is double-clicked.
<b>On Mouse Over</b>	pointer is moved over the element.
<b>On Mouse Leave</b>	pointer is moved out of the element.
<b>On Startup</b>	element is first shown in WindowViewer.
<b>While Mouse Over</b>	pointer is over the element.

**Note:** To expand the available space for your script, you can use the **Expansion** buttons to hide the script header and/or the Animation list.

### To configure an element with an action script animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Action Scripts**. The action scripts animation is added to the Animation list and the **Action Scripts** configuration panel appears.
4. From the **Trigger type** list, select the trigger that activates the action script at run time.
5. If you select a trigger type that starts with "While", type how frequently the action script is executed at run time in the **Every** box.

6. If you select the trigger types **On Mouse Over** or **On Mouse Leave**, the **Every** box label shows **After** instead. Type a value in the **After** box. This value specifies after what delay the action script is executed at run time.
7. Specify a trigger type that involves pressing a key is run by typing a key or key combination. In the **Shortcut** area. Select a shortcut key in the **Key** list. Select **Ctrl** and/or **Shift** to combine the shortcut key with the Ctrl key and/or Shift key.
8. Create your script in the action script window.
9. Click **OK**.

## Configuring an Action Script Animation with a "Mouse-Down" Event Trigger

Action scripts that are activated with a "mouse-down" event (for example, On Left Click/Key/Touch Down) trigger two separate events:

- A mouse-down event, when the button is pressed.
- A mouse-up event, when the button is released.

As a result, timing issues can occur in if the user holds the button, even momentarily, before releasing it. These timing issues can affect how a pushbutton graphic is displayed. The pushbutton image may continue to show as depressed (active state), even after the user has released the button.

You can avoid this potential timing issue by using a "mouse-up" trigger instead of the "mouse down." A "mouse-up" trigger only sends one event (when the button is released), thus eliminating any timing issues that could result from a delay in the user releasing the button.

## Configuring a Show Symbol Animation

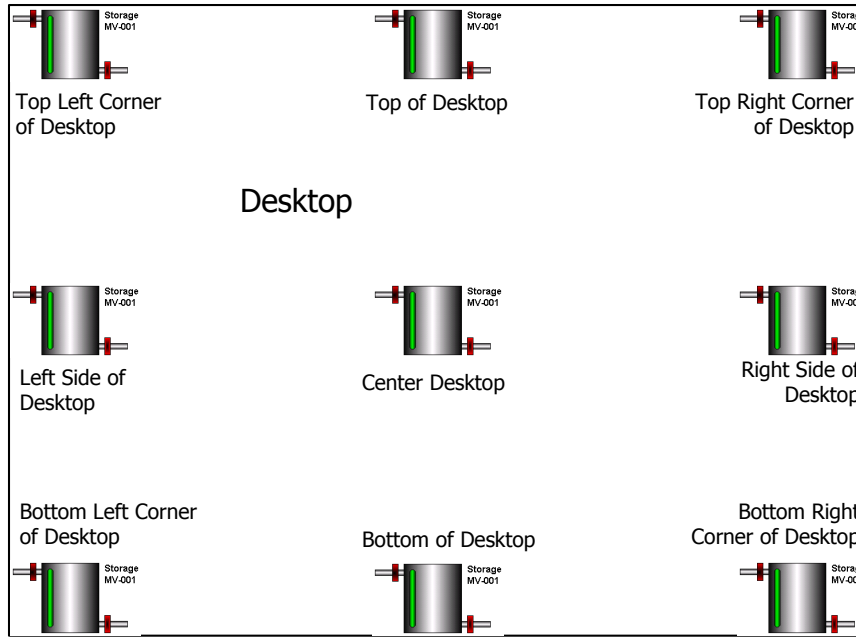
You can configure an element with a show symbol animation. A Show symbol animation shows a specified graphic in a new dialog box, when the element is clicked on.

You can configure:

- Which graphic appears in the new window.
- If the window has a title bar, and if so if it has a caption.
- If the window is modal or modeless.
- If the window has a close button.
- If the window can be resized.
- The initial window position.
- The size of the window.

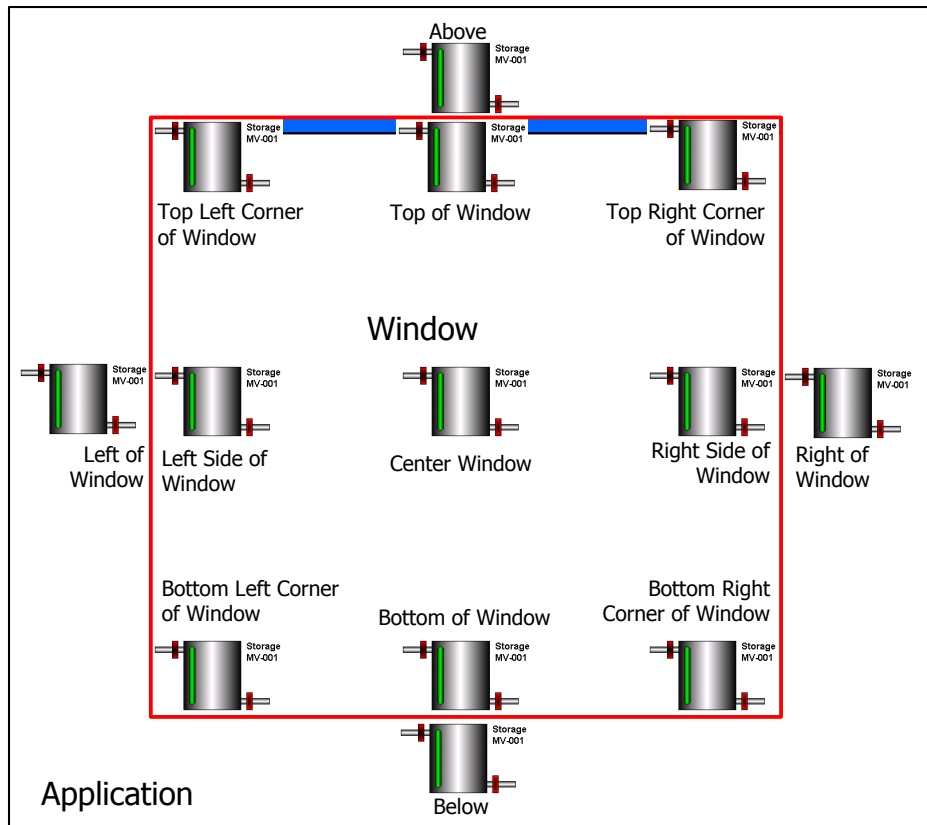
You can configure the position to be in relation of the:

- Desktop, such as at edges, corners, or at center.



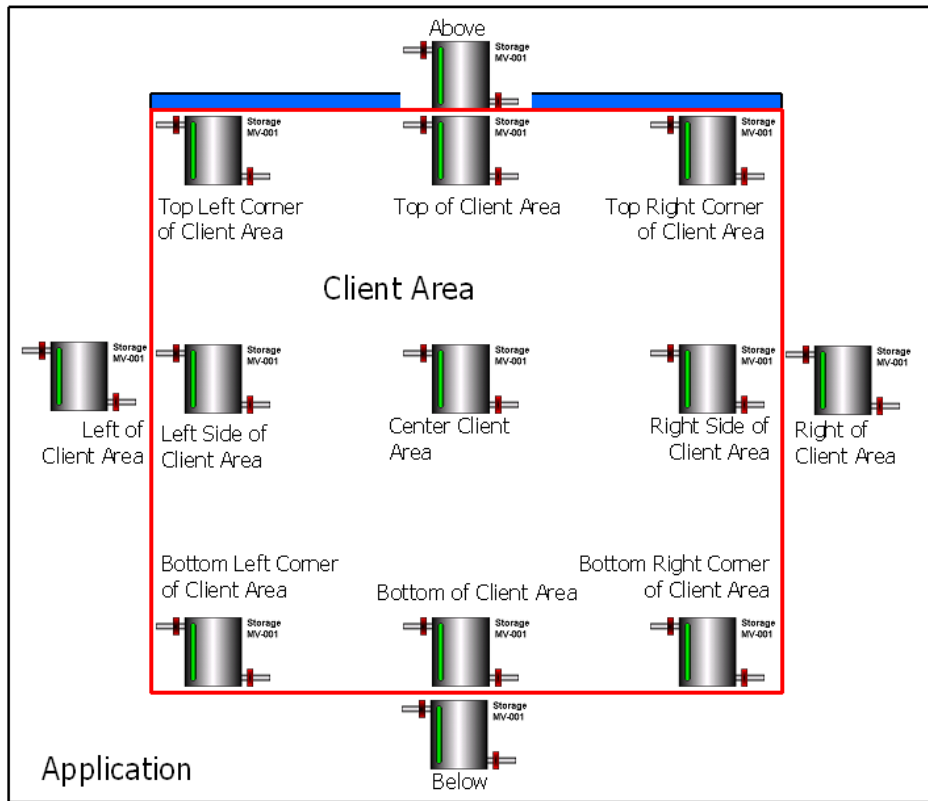
Positioning Related to Desktop Edges and Center

- Window, such as at one of its edges, its corners, its center or above, below, to the left or right. The window area includes the title bar if it appears.



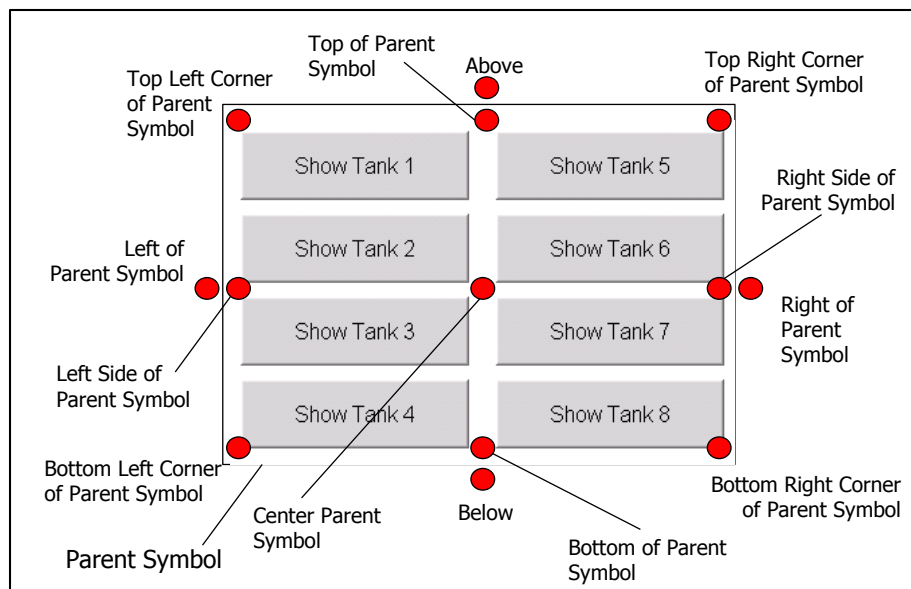
Positioning Related to Window

- Client Area of your HMI/SCADA software.



Positioning Related to Client Area

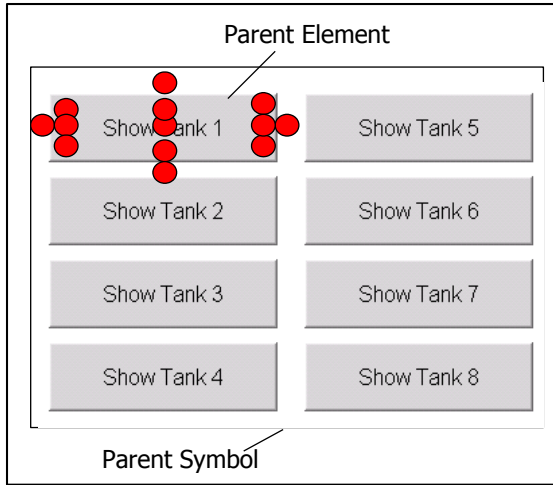
- Source Symbol, in which case the Show Symbol window is positioned in relation to the entire source symbol that contains the element that called the window.



Positioning Related to Parent Symbol

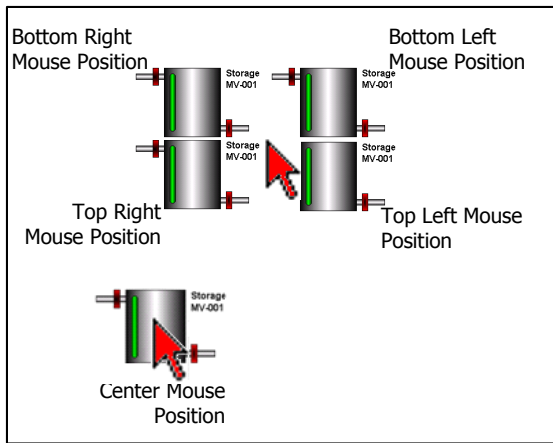


- Parent Element, in which case the Show Symbol window is positioned in relation just to the element that called the Show Symbol window.



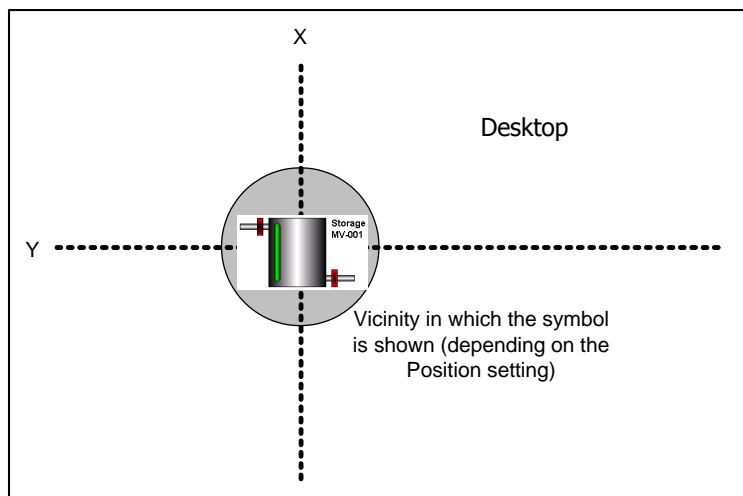
Positioning Related to Parent Element

- Mouse, in which case the Show Symbol window is positioned in relation to the pointer coordinates.



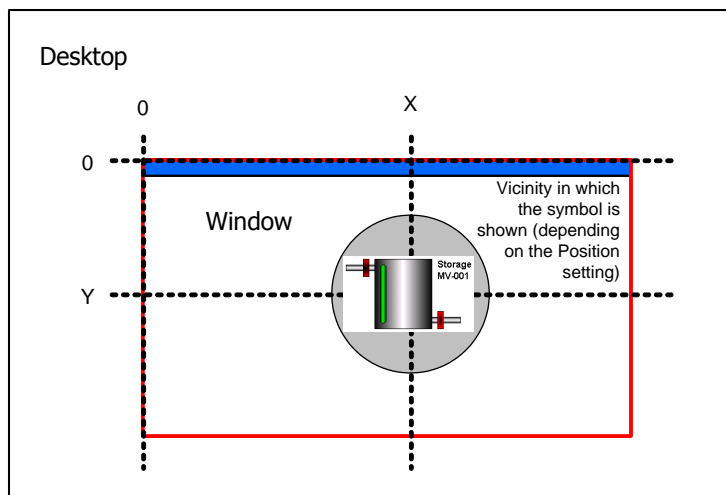
Selected Positionings Related to Mouse Pointer

- Desktop coordinates. The graphic is placed in the vicinity of coordinates that relate to the desktop.



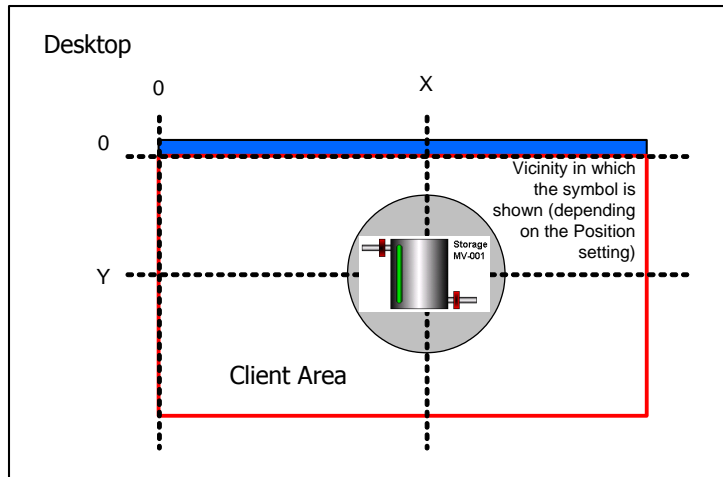
X, Y Positioning Related to Desktop

- Window coordinates. The graphic is placed in the vicinity of coordinates that relate to the window, including the title bar if shown.



X, Y Positioning Related to Window

- Client Area coordinates. The graphic is placed in the vicinity of coordinates that relate to the client area.



X, Y Positioning Related to Client Area

**To configure an element with a show symbol animation**

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Show Symbol**. The Show Symbol animation is added to the Animation list.
4. Configure the graphic. Do one or more of the following:
  - a. In the **Reference** box, type a graphic name or browse for one by using the browse button.
  - b. To add a title bar to the graphic, select **Has Title Bar**.
  - c. To use the graphic name as window title, select **Use Symbol Name for Window Title**.
  - d. Select the window type, **Modal** or **Modeless**.
  - e. To add a Close button, select **Has Close Button**.
  - f. To add resize controls, select **Resizable**.
5. Select where you want the window to appear by selecting a position in the **Position** lists. The first list contains positions that are in relation to the item of the second list. Select one of the following:
  - Center**
  - Top Left Corner**
  - Top Right Corner**
  - Left**
  - Right of**
  - Lower**
  - Below**
  - Above**
  - Top**
  - Left of**
  - Right Side**

- Lower Left Corner**
- Lower Right Corner**

From the second list, select the item the position is referring to:

- Desktop:** relative to the entire desktop
  - Window:** relative to the window.
  - Client Area:** relative to the client area.
  - Parent Symbol:** relative to the entire graphic that calls it.
  - Parent Element:** relative to the element or element group that calls it.
  - Mouse:** relative to the pointer.
  - Desktop X,Y:** relative to a specified coordinate on the desktop.
  - Window X,Y:** relative to a specified coordinate of the window.
  - Client Area X,Y:** relative to a specified coordinate of the client area.
6. If you select **Desktop X,Y** or **Window X,Y** or **Client Area X,Y** as position, type the new coordinates in the **X** and **Y** value boxes.
7. Select how large you want the window to be in the **Size** list. You can select:
- Relative to Symbol** to make the window size the same as the size of the graphic.
  - Custom Width and Height** to specify a width and height.
- Depending on your selection of the item the graphic is referring to, you can select:
- Relative to Desktop** to adjust the window size relative to the size of the desktop.
  - Relative to Window** to adjust the window size relative to window that contains the graphic that calls it.
  - Relative to Client Area** to adjust the window size relative to the client area.
  - Relative to Parent Symbol** to adjust the window size relative to the size of the graphic that calls it.
  - Relative to Parent Element** to adjust the window size relative to the size of the element that calls it.
8. Continue specifying position information.
- a. If you select **Relative...** as size, enter a scaling percentage in the **Scale Symbol** box.
  - b. If you select **Custom Width and Height** as size, type the new width and height in the **W** and **H** boxes.
  - c. If you select **Desktop, Window, Client Area, Parent Symbol** or **Parent Element** as referred item, you can configure the object to be stretched horizontally or vertically. Do one or both of the following:
    - Select **Stretch symbol to fit ... width** and enter a height in the **H** box.
    - Select **Stretch symbol to fit ... height** and enter a width in the **W** box.
9. You can specify that the graphic window appears by pressing a key or key combination. In the **Shortcut** area:
- a. Select a shortcut key in the **Key** list.
  - b. Select **Ctrl** and/or **Shift** to combine the shortcut key with the CTRL key and/or SHIFT key.
10. Click **OK**.

## Configuring a Hide Symbol Animation

You can configure an element with a Hide Symbol animation. The Hide Symbol animation lets you close:

- The current graphic
- A graphic that is shown by a specified element.

### To configure an element with a Hide Symbol animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Add** icon and select **Hide Symbol**. The Hide Symbol animation is added to the Animation list and the **Symbol Hide** configuration panel appears.
4. Select:
  - **Current Symbol** if you want to close the currently shown graphic.
  - **Symbol shown by an element** if you want to close a graphic that appears by that element. Type the element name in the adjacent box.
5. You can specify that the graphic window closes by pressing a key or key combination. In the **Shortcut** area:
  - a. Select a shortcut key in the **Key** list.
  - b. Select **Ctrl** and/or **Shift** to combine the shortcut key with the Ctrl key and/or Shift key.
6. Click **OK**.

## Configuring a Hyperlink Animation

You can configure an element with a hyperlink animation, i.e. a link when clicked will launch the default browser with a customizable URL. The animation will allow you to construct a URL animation using static text, reference or a compound expression.

### To configure an element with a hyperlink animation

1. Select the element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animation** dialog box appears.
3. Click the **Add** icon and select **Hyperlink**. The hyperlink animation is added to the Animation list and the **Hyperlink** configuration panel appears.
4. The hyperlink can be specified as static text, an expression, or a reference.
  - Select the T icon to indicate that it is a static value.
  - Select the tag icon to indicate that it is a reference to a value.
5. For a static value type the complete URL. For example: `www.google.com`
6. For a reference or expression value, open your HMI's attribute/tag browser and select a data source.
7. You can also form expression that will use tag values in runtime to generate a URL. For example: `"www.google.com/search?q=" + CPSearchText`
8. Select the **Protocol**; http or https.
9. You can specify a shortcut by pressing a key or key combination. In the **Shortcut** area:
  - Select a shortcut key in the **Key** list.
  - Select **Ctrl** and/or **Shift** to combine the shortcut key with the Ctrl key and/or Shift key.

10. If the value is set as static text, then you can click the **Preview** button and test the link.
11. Click **OK**.

## Configuring Element-Specific Animations

Some elements have their own unique animation type that can only be used for that element type. You cannot remove their unique animation, but depending on the element you can add and remove other common animations.

The elements with specific animations are:

- Status element
- Windows common controls

## Configuring Animation for a Status Element

You can configure the Status element with a DataStatus animation to indicate quality and status from:

- Attributes and tags used in elements with animation.
- Attributes and tags directly.

The appearance of the Status element depends on the settings in the **Configure Quality and Status Display** dialog box. For more information, see *Configuring Animation for a Status Element* on page 198.

The DataStatus animation is only used by the Status element and cannot be removed from the Status element.

### To configure a DataStatus animation

1. Select the Status element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears, showing the DataStatus configuration panel.
3. In the **Available Graphic Elements** list, select all elements for which you want to monitor their attribute quality and status.
4. Click the >> button to add them to the **Selected Graphic Elements** list.
5. Click the **Expression** tab. The **Expression** panel appears.
6. In the **Value or Expression** list, type a value or expression that can be a literal, or a reference or element property.

---

**Tip:** You can also browse for the reference by clicking the browse button.

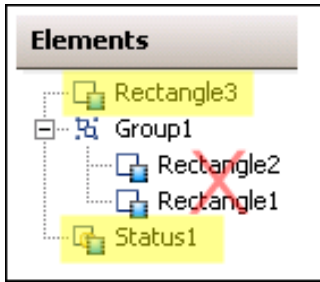
---

7. To add more values or expressions, click the **Add** button. An additional row is added for data input.
8. Click **OK**.

## Restrictions of the Status Element

The Status element must be in the same hierarchical level as the animated elements with the attributes you want to monitor.

If you move elements out of their hierarchical level after you associate them with a Status element, for example, by grouping them, their attributes are no longer monitored.



To avoid this problem, move a new Status element in the hierarchical level you want to monitor, or associate it directly with the attributes you want to monitor.

## Configuring a Radio Button Group Animation

The Radio Button Group animation is only used by the Radio Button Group element.

You can create a:

- **Static** radio button group - uses static captions and values that you define in the configuration panel.
- **Array** radio button group - uses captions and values contained in an AutomationObject array.
- **Enum** radio button group - uses captions and values contained in an enum data type of an AutomationObject.

## Configuring a Static Radio Button Group Animation

You can configure a radio button group with static values and captions.

### To configure a static radio button group animation

1. Select the radio button group element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears and the **Static Radio Button Group** configuration panel appears on the right side.
3. In the **Reference** box, type an attribute reference that is to be tied to the selected value at run time.  
You can select when to submit the value changes. For more information, see *Submitting the Value Changes* on page 209.
4. In the **Static Values and Captions** list, configure the captions of the radio button group and also the values that correspond to them to:
  - **Add an option** - click the **Add** icon.
  - **Delete an option** - select it in the list and click the **Remove** icon.
  - **Move an option up** the list - select it in the list and click the **Arrow up** icon.
  - **Move an option down** the list - select it in the list and click the **Arrow down** icon.
5. To use the values themselves as captions, select **Use Values as Captions**.
6. Orientate the radio button group in vertical or horizontal direction. Select **Vertical** or **Horizontal**.
7. Click **OK**.

## Configuring an Array Radio Button Group Animation

You can configure a radio button group with values from an array and captions.

### To configure an array radio button group animation

1. Select the radio button group element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Array** button. The **Array Radio Button Group** configuration panel appears on the right side.
4. In the **Reference** box, type an attribute reference that is to be tied to the selected value at run time. You can select when to submit the value changes. For more information, see *Submitting the Value Changes* on page 209.
5. In the **Array Reference** box, type or browse for an array attribute. The **Array Values and Captions** list shows the values from the array reference.
6. To define your own captions, clear **Use Values as Captions** and type them in the list.
7. To format the value before it appears as a caption, type a text format string in the **Format** box, for example **#.###**. Preceding zeroes are ignored if the array data type is numeric.
8. Set **Items Sorting** to:
  - **None** to show the items in the order they are in the array attribute.
  - **Ascending** to show the items sorted in ascending order.
  - **Descending** to show the items sorted in descending order.
9. Orientate the radio button group in vertical or horizontal direction. Select **Vertical** or **Horizontal**.
10. Click **OK**.

For example, you want to create a Radio Button Group in your symbol with the following options. The values to be written to the target attribute are contained in the user-defined attribute array called **Options** of an AutomationObject called **UD**.

Option	Value to be written
Open	1
Close	2
Hold	3
Report Error	4
Unknown	99

## Configuring an Enum Radio Button Group Animation

You can configure a radio button group with values from an enum attribute and captions.

### To configure an enum radio button group animation

1. Select the radio button group element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Enum** button. The **Enum Radio Button Group** configuration panel appears on the right side.
4. In the **Enum Reference** box, type an enum attribute reference. The **Enum Values and Captions** list shows the values from the enum reference.



You can select when to submit the value changes. For more information, see *Submitting the Value Changes* on page 209.

5. To define your own captions, clear **Use Values as Captions** and type them in the list.
6. Set **Items Sorting** to:
  - **None** to show the items in the order they are in the enum attribute.
  - **Ascending** to show the items sorted in ascending order.
  - **Descending** to show the items sorted in descending order.
7. Orientate the radio button group in vertical or horizontal direction. Select **Vertical** or **Horizontal**.
8. Click **OK**.

## Configuring a Check Box Animation

The Check Box animation is only used by the Check Box element.

### To configure a Check Box animation

1. Select the Check Box element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears and the **Check Box** configuration panel appears on the right side.
3. In the **Checked value - Boolean** box, type an attribute reference. The attribute reference is tied to the selected state of the check box control at run time.

You can select when to submit the value changes. For more information, see *Submitting the Value Changes* on page 209.

4. To set the caption of the check box at run-time, select **Override caption at Runtime with the following expression** and type a string value or attribute reference or expression in the **String Expression** box.
5. Click **OK**.

## Configuring an Edit Box Animation

The Edit Box animation is only used by the Edit Box element. You cannot remove this animation from the Edit Box element, but you can add certain common animations.

You can also use Edit Box-specific methods in scripting to get and set the text at run time. You can browse these methods in your HMI's attribute/tag browser with the Edit Box selected. For more information on these methods, see *Configuring Edit Box Methods* on page 223.

### To configure an Edit Box animation

1. Select the Edit Box element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears and the **Edit Box** configuration panel appears on the right side.
3. In the **String Reference** box, type an string attribute reference. The string attribute reference is tied to the text in the edit box at run time.

You can select when to submit the value changes. For more information, see *Submitting the Value Changes* on page 209.

---

**Tip:** Use the **On Trigger Condition** option to set when the Edit Box element writes the run-time value to the reference. This avoids a conflict between the run-time value of the Edit Box and run-time value of the reference.

---

4. In the **Configuration** area, select:

- **Multiline** to wrap the text into multiple lines in the edit box.
  - **Read-Only** to use the edit box to only show text and not allow text input.
  - **Maximum Length** to limit the maximum numbers of characters you can type in the edit box control. You can specify the maximum number in the **Characters** box.
5. Enter a default text in the **Text** box.

## Configuring a Combo Box Animation

The Combo Box animation is only used by the Combo Box element.

You can create a:

- **Static** combo box - uses static captions and values that you define in the configuration panel.
- **Array** combo box - uses captions and values contained in an array if supported by your HMI/SCADA software.
- **Enum** combo box - uses captions and values contained in an enum data type if supported by your HMI/SCADA software.

You can also use Combo Box-specific methods in scripting to perform various functions at run time. You can browse these methods in your HMI's attribute/tag browser with the Combo Box selected.

For more information on these methods, see *Configuring Combo Box and List Box Methods* on page 224.

## Configuring a Static Combo Box Animation

You can configure a combo box with static values and captions.

### To configure a static combo box animation

1. Select the combo box element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears and the **Static Combo Box** configuration panel appears on the right side.
3. In the **Reference** box, type an attribute reference that is to be tied to the selected value at run time.  
You can select when to submit the value changes. For more information, see *Submitting the Value Changes* on page 209.
4. In the **Static Values and Captions** list, configure the captions of the combo box and also the values that correspond to them:
  - Add an option** - click the **Add** icon.
  - Delete an option** - select it in the list and click the **Remove** icon.
  - Move an option up** the list - select it in the list and click the **Arrow up** icon.
  - Move an option down** the list - select it in the list and click the **Arrow down** icon.
5. Specify how you want to use captions. Do one of more of the following:
  - To use the values themselves as captions, select **Use Values as Captions**.
  - To alphabetically sort the captions, select **Sorted**.
  - To enable duplicate captions, select **Allow Duplicates**.

---

**Note:** If you clear **Allow Duplicates** and click **OK**, all duplicate captions are removed from combo box on the canvas. The captions are case-insensitive, so that for example "item1" is considered a duplicate of "Item1". The removal of the duplicate items is reflected when you re-open the **Edit Animations** dialog box.

---

6. Select the type of combo box from the **Type** list. Select:
  - Simple** - at run time you can type a value, or select one by using arrow up and arrow down buttons. However, you cannot see the list of values.
  - DropDown** - at run time you can type a value, or select one from the list.
  - DropDownList** - at run time you can only select a value from the list, but not type one.
7. Click **OK**.

## Configuring an Array Combo Box Animation

You can configure a combo box with values from an array and captions.

### To configure an array combo box animation

1. Select the combo box element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Array** button. The **Array Combo Box** configuration panel appears on the right side.
4. In the **Reference** box, type an attribute reference that is to be tied to the selected value at run time. The **Array Values and Captions** list shows the values from the array reference.

You can select when to submit the value changes. For more information, see *Submitting the Value Changes* on page 209.
5. To define your own captions, clear **Use Values as Captions** and type them in the list.
6. If you want to format the value before it appears as a caption, type a text format string in the **Format** box, for example **#.###**. Preceding zeroes are ignored if the array data type is numeric.
7. Set **Items Sorting** to:
  - None** to show the items in the order they are in the array attribute.
  - Ascending** to show the items sorted in ascending order.
  - Descending** to show the items sorted in descending order.
8. Click **OK**.

## Configuring an Enum Combo Box Animation

You can configure a combo box with values from an enum attribute and captions.

### To configure an enum combo box animation

1. Select the combo box element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Enum** button. The **Enum Combo Box** configuration panel appears on the right side.
4. In the **Enum Reference** box, type an enum attribute reference. The **Enum Values and Captions** list shows the values from the enum reference.

You can select when to submit the value changes. For more information, see *Submitting the Value Changes* on page 209.
5. To define your own captions, clear **Use Values as Captions** and type them in the list.
6. Set **Items Sorting** to:
  - None** to show the items in the order they are in the enum attribute.
  - Ascending** to show the items sorted in ascending order.
  - Descending** to show the items sorted in descending order.

7. Click **OK**.

## Configuring a Calendar Control Animation

The Calendar Control animation is only used by the Calendar Control element. The Calendar Control date format depends on the regional settings of the operating system.

### To configure a Calendar control animation

1. Select the Calendar control element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears and the **Calendar** configuration panel appears on the right side.
3. In the **Date Reference** box, type a Time attribute reference that is to be tied to the selected value at run time.

You can select when to submit the value changes. For more information, see *Submitting the Value Changes* on page 209.

4. To restrict the date the user can select at run time, specify limits as follows:
  - In the **MinDate** box, type a lower limit for the date.
  - In the **MaxDate** box, type an upper limit for date.
5. To show some dates as bold, in the **Bolded Dates** box, type a reference that points to an attribute array with time data type.
6. To show today's date on the calendar control, select **Show Today**.
7. To change the colors of the calendar control, click in the **Calendar Colors** area the following color boxes:
  - **Month Background**
  - **Month Trailing Forecolor**
  - **Title Background**
  - **Title Foreground**The **Select FillColor** dialog box appears and you can select a solid color.
8. Click **OK**.

## Configuring a DateTime Picker Animation

The DateTime Picker animation is only used by the DateTime Picker element.

### To configure a DateTime Picker animation

1. Select the DateTime Picker control element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears and the **DateTime Picker** configuration panel appears.
3. In the **Time Reference** box, type a Time attribute reference that is to be tied to the selected value at run time.

You can select when to submit the value changes. For more information, see *Submitting the Value Changes* on page 209.

4. To set the datetime format, select one of the following from the **Format** list:
  - Long** to show the date and time in the long format of the operating system, for example: Thursday, August 03 2006.

- **Short** to show the date and time in the short format of the operating system, for example: 8/3/2006.
- **Time** to show just the time in the time format of the operating system, for example: 3:46:09 PM.
- **Custom** to specify your own time format. Use the following letters to set the time format:
  - h One or two-digit hour in 12-hour format.
  - hh Two-digit hour in 12-hour format. Single digit values are preceded by a zero.
  - H One or two-digit hour in 24-hour format.
  - HH Two-digit hour in 24-hour format. Single digit values are preceded by a zero.
  - t One-letter AM/PM abbreviation ("AM" appears as "A").
  - tt Two-letter AM/PM abbreviation ("AM" appears as "AM").
  - m One or two-digit minute.
  - mm Two-digit minute. Single digit values are preceded by a zero.
  - s One or two-digit seconds.
  - ss Two-digit seconds. Single digit values are preceded by a zero.
  - d One or two-digit day.
  - dd Two-digit day. Single digit day values are preceded by a zero.
  - ddd Three-character day-of-week abbreviation.
  - dddd Full day-of-week name.
  - M One or two-digit month number.
  - MM Two-digit month number. Single digit values are preceded by a zero.
  - MMM Three-character month abbreviation.
  - MMMM Full month name.
  - y One-digit year (2001 appears as "1").
  - yy Last two digits of the year (2001 appears as "01").
  - yyyy Full year (2001 appears as "2001").

---

**Note:** You can use any other characters, except "g" in the property. These characters then appear at design time and run time in the control.

---

5. To restrict the date the user can select at run time, you can specify limits in the:
  - **MinDate** box - type a lower limit for the date.
  - **MaxDate** box - type an upper limit for date.
6. To change the colors of the calendar control that drops down, click in the **Calendar Colors** area the following color boxes:

- Month Background**
- Month Trailing Forecolor**
- Title Background**
- Title Foreground**

The **Select FillColor** dialog box appears and you can select a solid color.

## Configuring a List Box Animation

The List Box animation is only used by the List Box element.

You can create a:

- **Static** list box - uses static captions and values that you define in the configuration panel.
- **Array** list box - uses captions and values contained in an array if supported by your HMI/SCADA software.
- **Enum** list box - uses captions and values contained in an enum data type if supported by your HMI/SCADA software.

You can also use List Box-specific methods in scripting to perform various functions at run time. You can browse these methods in your HMI's attribute/tag browser with the List Box selected.

For more information on these methods, see *Configuring Combo Box and List Box Methods* on page 224.

## Configuring a Static List Box Animation

You can configure a list box with static values and captions.

### To configure a static list box animation

1. Select the list box element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears and the **Static List Box** configuration panel appears on the right side.
3. In the **Reference** box, type an attribute reference that is to be tied to the selected value at run time.  
You can select when to submit the value changes. For more information, see *Submitting the Value Changes* on page 209.
4. In the **Static Values and Captions** list, configure the captions of the list box and also the values that correspond to them. To:
  - **Add an option** - click the **Add** icon.
  - **Delete an option** - select it in the list and click the **Remove** icon.
  - **Move an option up** the list - select it in the list and click the **Arrow up** icon.
  - **Move an option down** the list - select it in the list and click the **Arrow down** icon.
5. Specify how you want to use captions. Do one of more of the following:
  - If you want to use the values themselves as captions, select **Use Values as Captions**.
  - If you want to alphabetically sort the captions, select **Sorted**.
  - If you want to allow duplicate captions, select **Allow Duplicates**.
6. Click **OK**.

## Configuring an Array List Box Animation

You can configure a list box with values from an array and captions.

### To configure an array list box animation

1. Select the list box element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Array** button. The **Array List Box** configuration panel appears on the right side.
4. In the **Reference** box, type an attribute reference that is to be tied to the selected value at run time.  
You can select when to submit the value changes. For more information, see *Submitting the Value Changes* on page 209.
5. In the **Array Reference** box, type or browse for an array attribute. The **Array Values and Captions** list shows the values from the array reference.
6. To define your own captions, clear **Use Values as Captions** and type them in the list.
7. To format the value before it appears as a caption, type a text format string in the **Format** box, for example **###**. Preceding zeroes are ignored if the array data type is numeric.
8. Set **Items Sorting** to:
  - **None** to show the items in the order they are in the array attribute.
  - **Ascending** to show the items sorted in ascending order.
  - **Descending** to show the items sorted in descending order.
9. Click **OK**.

## Configuring an Enum List Box Animation

You can configure a list box with values from an enum attribute and captions.

### To configure an enum list box animation

1. Select the radio button group element.
2. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
3. Click the **Enum** button. The **Enum List Box** configuration panel appears on the right side.
4. In the **Enum Reference** box, type an enum attribute reference. The **Enum Values and Captions** list shows the values from the enum reference.
5. You can select when to submit the value changes. For more information, see *Submitting the Value Changes* on page 209.
6. To define your own captions, clear **Use Values as Captions** and type them in the list.
7. Set **Items Sorting** to:
  - **None** to show the items in the order they are in the enum attribute.
  - **Ascending** to show the items sorted in ascending order.
  - **Descending** to show the items sorted in descending order.
8. Click **OK**.

## Configuring a Trend Pen

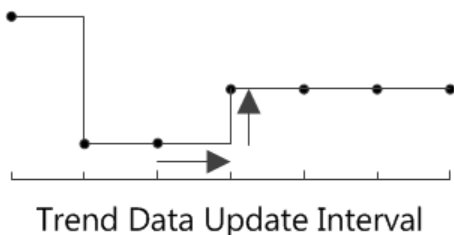
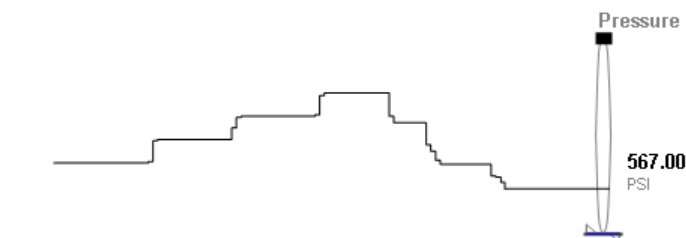
A Trend Pen shows a succession of process values as a trend line consisting of current and historical data updated at a minimum of one second intervals. A trend line gives operators a quick visual snapshot of a process value over a defined period.

### Understanding the Types of Trend Plots

You can configure two types of Trend Pen plots. A Line plot draws a line between each successive data point during the trend period.



A Step Line plot draws a horizontal line from a trend data point to the time of the next point on the trend's X-axis, and then draws a vertical line to the data point. A Step Line plot is the default for a Trend Pen.



### Understanding the Types of Trend Pen Periods

A trend time period is the interval of process values shown on the X-axis of the trend during run time, which consists of a start time, end time, and a duration.

- The StartTime and duration properties are read/write.
- The EndTime property is a read-only value that the system calculates by adding the duration to the start time.

You can configure two types of Trend Pen periods.

- Moving time period

In a Moving trend period, the start time of a trend period is the current time, and the end time is the duration of the time period from the start time. When the period ends, the next period begins. The start time for the next period is set to the end time of the previous trend period.

- Fixed time period



In a Fixed trend period, the start time is initially the current time. The start time of a trend period does not change automatically, but can be specified by a script using the StartTime property.

The end time of a Fixed trend period is set by the duration of the trend from the specified start time of the period.

$$\text{End Time} = \text{Start Time} + \text{Duration}$$

## Submitting the Value Changes

You can configure a Windows common control to write the data:

- Immediately when it is selected in the control at run time.
- When a specified Boolean expression becomes true.

---

**Note:** The Boolean expression is a trigger that determines when the value is written from the control to the tag or attribute. If the value changes in the tag or attribute, then the value is written to the control, regardless of the trigger setting or condition.

---

### To submit value changes immediately

1. Open the Windows common control in the **Edit Animations** dialog box.
2. In the **Submit Value Changes** area, select **Immediately**.

### To submit value changes after a Boolean expression becomes true

1. Open the Windows common control in the Edit Animations dialog box.
2. In the **Submit Value Changes** area, select **On Trigger Condition**.
3. In the **Boolean Expression** box, type a Boolean expression or browse for a Boolean attribute.

## Format Strings in Element-Specific Animations

Some element-specific animations support format strings that specify the format of data during run time when a graphic is displayed. Element specific -animations enable you to change the style of the displayed data without changing the graphic in the editor, either interactively through the use of static text or by referencing strings within data items, thereby making the format dynamic. A format string consists of one or more identifiers that define the output.

## Numbers

The following table lists the basic number formatting.

---

Identifier	Type	Format	Example Output for "1.42"	Example Output for "-12400"
c	Currency	{0:c}	\$1.42	-\$12,400
d	Decimal (whole number)	{0:d}	Error	-12400
e	Scientific	{0:e}	1.420000e+000	-1.240000e+004
f	Fixed point	{0:f}	1.42	-12400.00
g	General	{0:g}	1.42	-12400
n	Number with commas for thousands	{0:n}	1.42	-12,400
r	Round trip	{0:r}	1.42	Error

---

Identifier	Type	Format	Example Output for "1.42"	Example Output for "-12400"
x	Hexadecimal	{0:x4}	Error	cf90

The following table lists the custom number formatting.

Identifier	Type	Format	Example Output for "1500.42"	Notes
0	Zero placeholder	{0:00.0000}	1500.4200	Pads with zeroes.
#	Digit placeholder	{0:(#).##}	(1500).42	
.	Decimal point	{0:0.0}	1500.4	
,	Thousand separator	{0:0,0}	1,500	Must be between two zeroes.
.,	Number scaling	{0:0,.}	2	Comma adjacent to Period scales by 1000.
%	Percent	{0:0%}	150042%	Multiplies by 100, adds % sign.
e	Exponent placeholder	{0:00e+0}	15e+2	Many exponent formats available.
;	Group separator			Used to separate multiple formats in one string format (for example, including parentheses around a string if the value is negative; see <i>Format String Examples</i> on page 212).

## Dates

Date formatting is dependent on the system's regional settings; the example strings here are for the U.S. The following table lists the basic date formatting.

Identifier	Type	Example
d	Short date	10/12/2002
D	Long date	December 10, 2002
t	Short time	10:11 PM
T	Long time	10:11:29 PM
f	Full date and time	December 10, 2002 10:11 PM

Identifier	Type	Example
F	Full date and time (long)	December 10, 2002 10:11:29 PM
g	Default date and time	10/12/2002 10:11 PM
G	Default date and time (long)	10/12/2002 10:11:29 PM
M	Month day pattern	December 10
r	RFC1123 date string	Tue, 10 Dec 2002 22:11:29 GMT
s	Sortable date string	2002-12-10T22:11:29
u	Universal sortable, local time	2002-12-10 22:13:50Z
U	Universal sortable, GMT	December 11, 2002 3:13:50 AM
Y	Year month pattern	December, 2002

The following table lists the custom date formatting.

Identifier	Type	Format	Example Output
dd	Day	{0:dd}	10
ddd	Day name	{0:ddd}	Tue
dddd	Full day name	{0:dddd}	Tuesday
f, ff, ...	Second fractions	{0:fff}	932
gg, ...	Era	{0:gg}	A.D.
hh	2-digit hour	{0:hh}	10
HH	2-digit hour, 24-hr format	{0:HH}	22
mm	Minute 00-59	{0:mm}	38
MM	Month 01-12	{0:MM}	12
MMM	Month abbreviation	{0:MMM}	Dec
MMMM	Full month name	{0:MMMM}	December
ss	Seconds 00-59	{0:ss}	46
tt	AM or PM	{0:tt}	PM
yy	Year, 2 digits	{0:yy}	02
yyyy	Year	{0:yyyy}	2002
zz	Time zone offset, 2 digits	{0:zz}	-05
zzz	Full time zone offset	{0:zzz}	-05:00
:	Separator	{0:hh:mm:ss}	10:43:20

Identifier	Type	Format	Example Output
/	Separator	{0:dd/MM/yyyy}	10/12/2002

## Enumerations

Identifier	Type
g	Default (flag names if available, otherwise decimal)
f	Flags always
d	Integer always
x	Eight-digit hex

## Format String Examples

The following string is an example of a currency string:

```
String.Format("{0:$#,##0.00;($#,##0.00);Zero}", value);
```

This string example will output values as follows:

- **\$1,240.00** if passed **1243.50**.
- **(\$1,240.00)** if passed **-1243.50**.
- The string **Zero** if passed the number zero.

The following string is an example of a phone number string:

```
String.Format("{0:(###) ###-####}", 8005551212);
```

This string example will output **(800) 555-1212**.

## Cutting, Copying and Pasting Animations

You can cut, copy and paste animations and their configuration between different elements. This is useful when you want to duplicate the animations of one element such as a line, to a different type of element such as a polyline.

If you try to paste an animation to an element that is already configured with that animation, or does not support this animation, a message appears informing you why you cannot paste the animation.

### To copy and paste animations between elements

1. Select the element from which you want to copy the animations.
2. On the **Edit** menu, point to **Animations**, and then click **Copy**.
3. Select one or more elements to which you want to paste the animations.
4. On the **Edit** menu, point to **Animations**, and then click **Paste**. The animation links are copied from the source element to the target elements.

### To cut and paste animations between elements

1. Select the element from which you want to cut the animations.
2. On the **Edit** menu, point to **Animations**, and then click **Cut**.
3. Select one or more elements to which you want to paste the animations.

4. On the **Edit** menu, point to **Animations**, and then click **Paste**. The animation links are removed from the source element and copied to the target elements.

## Substituting References in Elements

You can search and replace references used by any element on your canvas. You can use:

- basic mode by replacing strings in a list.
- advanced functions such as find and replace, ignore or respect case-sensitivity and wildcards.

### To substitute references in a graphic by using the list

1. Select one or more elements.
2. Do one of the following:
  - Press **Ctrl + E**.
3. On the **Special** menu, click **Substitute References**.  
The **Substitute References** dialog box appears.
4. In the **New** column, type the reference to be replaced.
5. Click **OK**. All references are substituted accordingly in the elements.

### To substitute references in a graphic by using find and replace functions

1. Select one or more elements.
2. Do one of the following:
  - Press **Ctrl + E**.
  - On the **Special** menu, click **Substitute References**.  
The **Substitute References** dialog box appears.
3. Click **Find & Replace**. The dialog box expands and shows find and replace parameters.
4. Specify your find and replace options. Do one of more of the following:
  - To find specific references in the list, type a string in the **Find What** box and click **Find Next** to find the next string.
  - To replace a selected found string with another string, type a wstring in the **Replace with** box and click **Replace**.
  - To replace multiple references, type values in the **Find What** and **Replace with** boxes and click **Replace all**.
  - To specify the search is case-sensitive, select **Match Case**.
  - To find only entire words that match your search string, select **Match Whole Word Only**.
  - To use wildcards, select **Use Wildcards**. Valid wildcards are "\*" (asterisk) and "?" (question mark).  
 "\*" indicates any number of variable characters. For example, "s\*" to search for all strings starting with "s".  
 "?" indicates one single variable character. For example, "M\_7?t" to search for all strings that start with "M\_7" and end with "t" and have exactly 5 characters.
5. Click **OK**. All text strings are substituted accordingly in the elements.



# CHAPTER 9

## Adding and Maintaining Graphic Scripts

### About Graphic Scripts

You can associate scripts to your graphics. Scripts can add animation to a graphic or its elements that can be executed in run time.

---

**Caution:** If you configure scripts that affect more than element and graphic animation, the script processing may affect performance.

---

You can:

- Configure the predefined scripts of a graphic.
- Add named scripts to a graphic.
- Edit existing named or predefined scripts in a graphic.
- Rename named scripts in a graphic.
- Remove named scripts from a graphic.
- Substitute references in named or predefined scripts.
- Use element methods in named or predefined scripts.

The autocomplete feature is available in the Graphic Editor script editor.

### Predefined and Named Scripts

Predefined graphic scripts run:

- One time when the graphic is shown or opened: **On Show**
- Periodically while the graphic is showing: **While Showing**
- One time when the graphic is hidden or closed: **On Hide**
- Any combination of the above

Named graphic scripts enable any number of scripts to run that are triggered by values or expressions during runtime :

- Being true: **While True**
- Being false: **While False**
- Transitioning from false to true: **On True**
- Transitioning from true to false: **On False**
- Change in value and/or quality: **DataChange**

The name of named scripts can be up to 32 characters in length, contain at least one letter, and contain special characters, such as #, \$, and \_.

### Execution Order of Graphic Scripts

When the graphic is showing, the scripts run in the following order:

1. On Show script.
2. Named scripts, not necessarily in the order that they appear in the list.

Any named script that is triggered by the DataChange trigger type runs the first time when the reference is subscribed to. This behavior is different than the DataChange trigger behavior of Application Server scripts and can take considerable time in intermittent networks.

---

**Note:** A named script will not run if the script is triggered by the DataChange trigger type and is bound to an HMI tag whose quality is Initializing, or whose quality is Bad and category is not OK.

---

## Graphic Script Time outs (already in ITAAIntegration Guide)

To avoid infinite loops in a graphic script, a time-out limit can be set in which FOR loops must complete execution. If a script loop does not complete execution within the time-out limit, WindowViewer automatically terminates the loop and writes a message to the Logger.

The time-out limit is checked only at the NEXT statement of the loop. Therefore, the first iteration of the loop is always executed, even if it exceeds the time-out limit.

### To change the time out for a graphic script

1. In WindowMaker, on the **Special** menu, point to **Configure** and click **WindowViewer**. The **WindowViewer Properties** dialog box appears.
2. Click the **Managed Application** tab.
3. In the **Script timeout (msec)** box, type a time-out value in milliseconds. Valid values are from 1 to 360,000.
4. Click **OK**.

## Security in Graphic Scripts

If the graphic script attempts to write to attributes or tags with Secured Write or Verified Write security classification, the script execution stops and the authentication dialog box appears.

After you enter correct authentication information, the graphic script continues execution.

---

**Note:** This feature applies only if Secured and Verified Writes are supported by your HMI/SCADA software application.

---

## Error Handling

A graphic script does not run if it contains a syntax error. When the graphic is loaded, a message is written to the Logger.

## Signature Security for Acknowledging Alarms

SignedAlarmAck() is a script function for Industrial Graphics to acknowledge one or more alarms on attributes or tags that optionally require a signature depending on whether any of the indicated alarms falls within a designated priority range. If so, the user must perform an authentication of the operation to acknowledge the alarms.

For information about SignedAlarmAck() script function syntax, parameters, and to see script examples, see the *SignedAlarmAck()* on page 366.

---

**Note:** This feature requires that your HMI/SCADA software supports SignedAlarmAcks.

---



## SignedAlarmAck() Run-time Behavior

At run time, the SignedAlarmAck() function does the following:

1. The function checks whether a signature is required on the alarms to be acknowledged.
  - a. It checks if the parameter Signature\_Reqd\_for\_Range is true.
  - b. If so, it checks if security is enabled on the Galaxy.
  - c. If so, it checks the Priority for each designated alarm and compares it against the indicated priority range. If any of the designated alarms falls within the priority range, a signature is required.
  - d. If none of the designated alarms falls within the priority range, but no one is logged in, a signature is required.
  - e. If no alarm is waiting for an acknowledgement, the function will do nothing, but will return a value indicating that no alarms are waiting for acknowledgement.
2. If none of the indicated alarms requires a signature, the function displays a simple pop-up acknowledgement dialog.
  - a. When the user clicks **OK**, the function writes the acknowledgement comment to the AckMsg attribute of each of the alarms identified in the Alarm\_List parameter.

The system identifies the logged-on user, if any, as the one who acknowledged all of the alarms.
  - b. If the user has permission to acknowledged alarms, or the galaxy is unsecured, the alarms are marked as having been acknowledged.
  - c. If the acknowledgement fails, there is no direct feedback to the user. The status of the alarms, however, will show that they have not been acknowledged.
3. If any of the indicated alarms requires a signature and is waiting for an acknowledgement, the function displays a pop-up acknowledgement dialog that requires a signature.
  - a. This dialog has edit fields for the user's credentials: name, password, and domain. By default, the user displayed is the logged-in user, if any. Otherwise, it is blank. All of these fields can be edited.

If Smart Cards are not enabled, the mode buttons for selecting Smart Card or password authentication are disabled.
  - b. The user enters the acknowledgement comment, if enabled, and the user's credentials.
  - c. The function validates the user's credentials.
  - d. If the user credentials are invalid the function displays an error message.

When the user clicks **OK** on the error message, the function re-displays the alarm authentication dialog, and allows the user to try again.

When the dialog is re-displayed, it shows the same user's name, domain, and acknowledgement comment as were entered, but the password or Smart Card PIN field is blank. The user may then re-try the authentication or cancel.
  - e. If the user credentials are valid, the function writes the acknowledgement comment to the AckMsg attribute of each of the alarms identified in the Alarm\_List parameter.

The system identifies the authorizing user as the one who acknowledged all of the alarms, including those that do not require a signature.

If the user has permission to acknowledgement alarms, the alarms are marked as having been acknowledged.

If the acknowledgement fails, there is no direct feedback to the user. The status of the alarms, however, will show that they have not been acknowledged.

4. The function provides a return value, and writes an information message in the Logger if an error occurs or the operation is canceled.

For information on run-time behavior and the sequence of executing script operations, see SignedWrite() Script Execution Sequence at Run Time.

## SignedAlarmAck() Scripting Tips

### SignedAlarmAck() and Alarm Configuration

You can use the SignedAlarmAck() function only in client scripts supported by your HMI/SCADA software.

### SignedAlarmAck() with OnShow and OnHide Scripts

We do not recommend using the SignedAlarmAck() function with OnShow and OnHide scripts. This can cause issues with window functionality, including the window title bar, windows losing correct focus, and windows opening on top of one another.

### SignedAlarmAck() with While True Scripts

We do not recommend using the SignedAlarmAck() function in a While True script type. A signed alarm acknowledgement requires user interaction. If you want to use a While True type script, it must be set to an execution time of 30-seconds or longer to allow the user to enter the required information.

## SignedAlarmAck() Applied Example

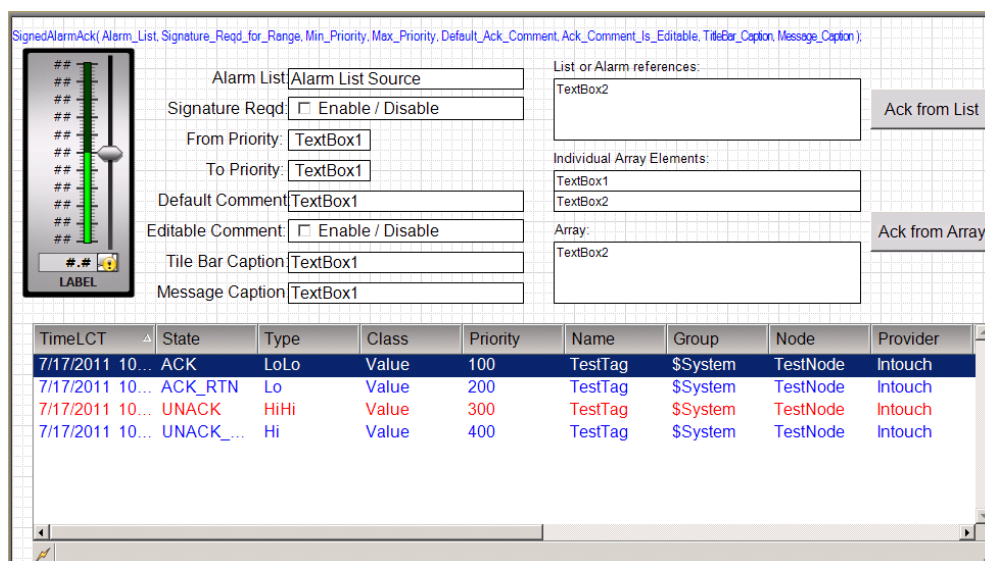
You can create a dashboard application to automate routine use of the SignedAlarmAck() function.

### To configure signature required for alarm acknowledgement

1. Use the Industrial Graphic Editor to create a graphic. Embed the graphic elements you require, such as buttons, Alarm Control, alarm acknowledgement and commenting configuration, and comment text boxes. Examples are shown in the following illustration.

If you embed an Alarm Control, as shown in this example, enable **Requires ACK Signature** in the **Runtime Behavior** animation to require a signature for alarm acknowledgement.

2. Add the SignedAlarmAck() script function as required. The following Industrial Graphic Editor detail shows two buttons configured with action scripts:



3. Configure the scripted functionality you require. Scripts for the buttons shown in the example, plus scripts for other possible button functions, are as follows:

- a. The alarm list and all other parameters are hard-coded into the script function.

The following example requests acknowledgement of three alarms. If the alarms are within the priority range 1-500 an authentication signature will be required. The comment is disabled from the operator.

```
ReturnCode = SignedAlarmAck ("Tank1.TankLevel.Lo Tank1.TankLevel.Hi
Tank1.TankLevel.HiHi", True, 1, 500, "Ack Tank Level", False, "Acknowledge
Alarms", "Please Acknowledge the Tank Level Alarms");
```

- b. The alarm list is passed as a parameter to the script from a string type Custom Property.

The following example is the same as example 4a above, except that the alarm list is a parameter pointing to a string type Custom Property which has concatenated the names of the alarms.

```
ReturnCode = SignedAlarmAck (TankLevel_Alarm_List, True, 1, 500, "Ack Tank
Level", False, "Acknowledge Alarms", "Please Acknowledge the Tank Level
Alarms");
```

- c. The alarm list is passed as a parameter to the script from an Attribute which is an array of strings.

The following example is the same as example 4a above, except that the alarm list is a parameter pointing to a string type array Attribute which has each of the alarm names as an array element.

```
ReturnCode = SignedAlarmAck (DataUDO.StringArray[], True, 1, 500, "Ack
Tank Level", False, "Acknowledge Alarms", "Please Acknowledge the Tank
Level Alarms");
```

- d. The alarm list is passed as a parameter to the script from an Attribute which is an array of strings. All other parameters are passed as script variables.

```
ReturnCode = SignedAlarmAck( DataUDO.StringArray[], EnableAckSig,
FromPriority, ToPriority, Default_Ack_Comment, Comment_Is_Editable,
Title_Bar_Caption, Message_Caption);
```

## Configuring the Predefined Scripts of a Graphic

You can configure the predefined scripts of a graphic. The predefined scripts can consist of:

- A script that runs one time when the graphic opens (On Show).
- A script that runs periodically as long as the graphic is open (While Showing).
- A script that runs one time when the graphic closes (On Hide).

---

**Note:** The **Predefined Scripts** animation cannot be deleted. It can contain scripts for each trigger type **On Show**, **While Showing** and **On Hide**.

---

### To configure the predefined scripts for a graphic

1. Open the graphic in the Industrial Graphic Editor.
2. On the **Special** menu, click **Scripts**. The **Edit Scripts** dialog box appears.
3. In the **Trigger Type** list, click:
  - **On Show** to configure a script that runs one time when the graphic opens.

If you create an OnShow script that uses a custom property bound to a tag, there is no guarantee that the tag data is valid when the script runs. This could occur because of the asynchronous nature of data subscriptions in HMI/SCADA software. Your script should first test the quality and status of the tag value before it is used in the rest of the script.

When the **On Show** trigger type is selected, the field that appears to the right of the **Type** list becomes a **Data Timeout** field. For more information about using this field, see *Ensuring Proper OnShow Script Execution* on page 220.

- **While Showing** to configure a script that runs periodically while the graphic is open.
  - **On Hide** to configure a script that runs one time when the graphic closes.
4. If you configured a **While Showing** script, type a time period in milliseconds in the **Period** box. This specifies after how many milliseconds the action script is executed.

---

**Note:** If you set the **While Showing** period too low, system performance may decrease.

---

5. Type your script in the main edit box. The script syntax is the same as the syntax of AutomationObject scripting.

---

**Note:** If the graphic includes a custom property, the name of the custom property and a nested class property in the script cannot be the same.

---

6. Select external data using your HMI's browser or explorer.
7. When you are done, click **OK**. The script editor checks the syntax of the script and may inform you of invalid syntax. Click:
  - **Yes** to save changes even if the script contains errors.
  - **No** to cancel the changes and close the script dialog box.

## Ensuring Proper OnShow Script Execution

When an OnShow script includes external references to tags, it is possible that the data from these tags is not yet available when the OnShow script runs. As a result, the script might not work properly.

To avoid this situation, you can enter a value in the **Data Timeout** field. For the duration of the data time-out period, the system checks for the presence of the external reference data. After all external reference data is present, the system executes the OnShow script.

If the data time-out period expires before all external data is present, the OnShow script is executed. However, the script might not work properly.

The default value in the **Data Timeout** field is:

- For new Industrial Graphics, 1,000 ms.
- For some HMI symbols, 0 ms; that is, the presence of external reference data is not checked. Verify the behavior of your HMI/SCADA software.

The maximum data time-out value is 30,000 ms.

Note the following issues regarding OnShow scripts and the Data Timeout function:

- The Data Timeout function is not available for the other trigger script types. It would be rare for external reference data to not be available in time for those scripts.
- The execution of the OnShow script is not delayed if there is an invalid reference (that is, the reference's quality is Bad).
- Named scripts are blocked until the OnShow script has completed, so some could be missed. For example, the named script OnDataChange might not run for the first few updates.
- Delayed OnShow scripts within nested embedded graphics might run out of order for the different nested levels. If the outer-most level is delayed but the inner levels are not delayed and are executed immediately, the order of execution will be changed.

## Adding Named Scripts to a Graphic

You can add named scripts to a graphic. A named script runs:

- One time when the specified values, data or expressions change.
- Periodically if the values or expressions meet a certain criterion, such as being true.

Every named script can contain only one trigger type.

To add a named script to a graphic

1. Open the graphic in the Industrial Graphic Editor.
2. On the **Special** menu, click **Scripts**. The **Edit Scripts** dialog box appears.
3. Click the Add icon. A new entry is created in the **Named Scripts** list.
4. Type a name for the named script. The name appears on the right panel as header.
5. In the **Expression** box, do one of the following:
  - Type an expression, value or reference.
  - Browse for a reference.  
The expression acts as data source for the script trigger.
6. In the **Trigger** list, click:
  - **WhileTrue** to trigger the script periodically when the expression is true.
  - **WhileFalse** to trigger the script periodically when the expression is false.
  - **OnTrue** to trigger the script one time when the expression becomes true from false.
  - **OnFalse** to trigger the script one time when the expression becomes false from true.
  - **DataChange** to trigger the script one time when the expression or its quality changes. Select the **Quality Changes** check box to trigger the script when the quality of the specified expression changes.
7. If you want to specify how often the script is run when the trigger condition is fulfilled, type a time delay in milliseconds in the **Trigger Period** box.
8. If you want to specify by how much the evaluated value must change before the script runs, type a deadband value in the **Deadband** box.
9. Type your script in the main edit box.
10. Use the Script Function Browser and Attribute Browser to select external data.
11. Click **OK**. The script editor validates the syntax of the script and identifies any errors. Click:
  - **Yes** to save changes even if the script contains errors.
  - **No** to cancel the changes and close the script dialog box.

## Editing Graphic Scripts

You can edit predefined and named graphic scripts.

### To edit graphic scripts

1. Open the graphic in the Industrial Graphic Editor.
2. On the **Special** menu, click **Scripts**. The **Edit Scripts** dialog box appears.
3. Select the script from the list. The right pane shows the script configuration.
4. If you are editing a predefined script, select the script trigger from the **TriggerType** list:

- **On Show** if the action script you want to edit runs one time after the graphic opens.
  - **While Showing** if the action script you want to edit runs periodically while the graphic is open.
  - **On Hide** if the action script you want to edit runs one time when the graphic closes.
5. Edit the action script in the script box.
  6. Click **OK**.

## Renaming Scripts in a Graphic

You can rename named scripts in a graphic. When you rename the named script, the functionality of the script does not change.

### To rename a named script

1. Open the graphic in the Industrial Graphic Editor.
2. On the **Special** menu, click **Scripts**. The **Edit Scripts** dialog box appears.
3. In the **Named Scripts** list, click the script you want to rename.
4. Click the script again. It appears in edit mode.
5. Enter a new name for the script and click **Enter**. The script is renamed.

## Removing Scripts from a Graphic

You can remove predefined or named scripts from a graphic.

### To remove predefined scripts from a graphic

1. Open the graphic in the Industrial Graphic Editor.
2. On the **Special** menu, click **Scripts**. The **Edit Scripts** dialog box appears.
3. Select **Predefined Scripts** from the list.
4. In the **Trigger type** list, click:
  - **On Show** if the action script you want to remove runs one time after the graphic opens.
  - **While Showing** if the action script you want to remove runs periodically while the graphic is open.
  - **On Hide** if the action script you want to remove runs one time after the graphic closes.
5. Delete all the script content in the script box.
6. Click **OK**.

### To remove named scripts from a graphic

1. Open the graphic in the Industrial Graphic Editor.
2. On the **Special** menu, click **Scripts**. The **Edit Scripts** dialog box appears.
3. Select the named script from the list.
4. Click the Remove icon. A message appears.
5. Click **Yes**. The script is removed.

## Substituting Attribute References in Scripts

You can substitute attribute references in scripts in the same way as you would with attribute references in elements. For more information, see *Substituting References in Elements* on page 213.

## Example of Changing Element Properties using Scripts

You can change some properties of elements using scripting. This lets you configure additional run-time behavior to your elements in addition to design-time animation of those elements.

When you write scripts for the graphic or for one of its elements, you can use your HMI's attribute/tag browser to show and select a:

- Property of an element
- Custom property of the graphic

If a reference is not unique, the following order applies:

1. Dimensioned variable references
2. Graphic properties references
3. Custom property references
4. Object attribute references

### To select an element property or a graphic custom property

1. Open your HMI's attribute/tag browser. It shows the element names and the properties of the selected element.
2. Select an element or graphic from the list. The right pane shows the accessible properties of the selected element or graphic.
3. Select a property from the right pane and click **OK**. The reference appears in the script window.

## Using Methods in Scripting

Some elements, such as the Edit Box, Combo Box and List Box controls, support methods in scripting. These methods can be used to perform various functions on the elements themselves at run time.

You can see the properties and methods supported by any given element by opening your HMI's attribute/tag browser and selecting the element.

You can use the methods of the:

- Edit Box control to save and load the text at run time to and from a file.
- Combo Box and List Box controls to access and change the contents of their lists at run time.

## Configuring Edit Box Methods

You can use the methods of an Edit Box control to:

- Save the contained text at run time to a file.
- Load text into the control from a file at run time.

### To save the contained text in an Edit Box control

- In an action script, use the following method:

```
ControlName.SaveText (FileName) ;
```

where `ControlName` is the name of the Edit Box control and `FileName` is the name of the file in which to save the contents of the control.

The text contained in the control at run time is saved to the specified file.

If you only specify a file name, the file is saved by default in the user folder of the operating system. For example: `c:\documents and settings\username`.



### To load text into an Edit Box control from a file

- In an action script, use the following method:

```
ControlName.LoadText (FileName) ;
```

where `ControlName` is the name of the Edit Box control and `FileName` is the name of the file you want to load the text from.

The text contained in the file is loaded into the run time contents of the Edit Box control.

If you only specify a file name, by default, the file is expected to be in the user folder of the operating system. For example: `c:\documents and settings\username`.

## Configuring Combo Box and List Box Methods

The Combo Box and List Box controls have methods that you can use to access and change the items in the list at run time. Typically, you configure an action script to access these methods.

You can:

- Add and insert items into the list.
- Delete individual or all items from the list.
- Find an item in the list.
- Get the item caption based on a specified index.
- Associate the items with values.
- Load items from and save items to a file.

## Adding and Inserting Items into a List

You can add an individual item:

- To the end of the list.
- Above the currently selected item.

### To add an item to a Combo Box or List Box list

- In an action script, use the following method:

```
ControlName.AddItem ("ItemCaption") ;
```

where `ControlName` is the name of the Combo Box or List Box control and `ItemCaption` is the new item you want to add.

The item is added to the end of the list.

---

**Note:** You can specify an additional parameter `writeToSelectedItem` in the `.AddItem` function for Combo Box controls. If `writeToSelectedItem` is `false`, the newly added item is not written to the reference configured in the Combo box's *Selected Item Value*. For example:  
`Controlname.AddItem("ItemCaption", bool writeToSelectedItem);`

---

### To insert an item in a Combo Box or List Box list

- In an action script, use the following method:

```
Controlname.InsertItem ("ItemCaption") ;
```

where `ControlName` is the name of the Combo Box or List Box control and `ItemCaption` is the new item you want to insert.



The item is inserted above the currently selected item in the list.

## Deleting Items from a List

You can delete:

- An individual item from a list.
- The selected item from a list.
- All items from a list.

If items cannot be deleted from a list at run time, no warning message is shown. Such items include Combo Box and List Box controls configured with enums or arrays.

### To delete an individual item from a Combo Box or List Box list

- In an action script, use the following method:

```
ControlName.DeleteItem(Index);
```

where `ControlName` is the name of the Combo Box or List Box control and `Index` is the index of the item you want to delete. The first item of the list has an index of 0.

The item at the specified index is deleted, subsequent items are moved up the list.

### To delete the selected item from a Combo Box or List Box list

- In an action script, use the following method:

```
ControlName.DeleteSelection();
```

where `ControlName` is the name of the Combo Box or List Box control.

The selected item is deleted, subsequent items are moved up the list.

### To delete all items from a Combo Box or List Box list

- In an action script, use the following method:

```
ControlName.Clear();
```

where `ControlName` is the name of the Combo Box or List Box control.

All items of the control are deleted.

## Finding an Item in a List

You can find an item in a Combo Box or List Box list. You specify the item caption, and the method returns the index number of the first item found. Otherwise, the method returns -1.

### Finding an item in a Combo Box or List Box list

- In an action script, use the following method:

```
Index = ControlName.FindItem("ItemCaption");
```

where `ControlName` is the name of the Combo Box or List Box control and `ItemCaption` is the caption of the item you are looking for.

The index is set to -1 if the item is not found, otherwise it contains the index of the first found item. The first item of the list has an index of 0.

## Reading the Caption of a Selected Item in a List

You can read the caption of a selected item in a Combo Box or List Box list.

## Reading the caption of a selected item in a Combo Box or List Box list

- In an action script, use the following method:

```
Caption = ControlName.GetItem(Index);
```

where `ControlName` is the name of the Combo Box or List Box control. `Index` is the index of the item for which you want to read the caption. The first item of the list has an index of 0.

`Caption` contains the item caption of the specified index.

## Associating Items with Values in a List

You can associate items with values in a Combo Box or List Box control. This is the same as using a secondary index system to identify items in the list.

You can:

- Set item data, which associates an item with a value
- Get item data, which returns the value that is associated with an item

### To set item data in a Combo Box or List Box list

- In an action script, use the following method:

```
ControlName.SetItemData(Index, Value);
```

where `ControlName` is the name of the Combo Box or List Box control, `Index` is the index of the item that you want to set and `Value` is the value you want to assign to the item. The first item of the list has an index of 0.

### To get item data in a Combo Box or List Box list

- In an action script, use the following method:

```
Value = ControlName.GetItemData(Index);
```

where `ControlName` is the name of the Combo Box or List Box control and `Index` is the index of the item for which you want to get the value. The first item of the list has an index of 0.

`Value` contains the value that is assigned to the item.

## Loading and Saving Item Lists

You can load and save all items in a list from and to a file.

### To load the item list for a Combo Box or List Box control from a file

- In an action script, use the following method:

```
ControlName.LoadList(FileName);
```

where `ControlName` is the name of the Combo Box or List Box control and `FileName` is the name of a file on the local harddrive or on the network.

If you only specify a file name, the file is expected to be in the users folder. For example: `c:\documents and settings\username`.

The list contained in the file is loaded and, if valid, the current list is overwritten.

### To save the item list for a Combo Box or List Box control to a file

- In an action script, use the following method:

```
Controlname.SaveList(FileName);
```

where `ControlName` is the name of the Combo Box or List Box control and `FileName` is the name of a file on the local harddrive or on the network.

If you only specify a file name, the file is saved to the users folder. For example, c:\documents and settings\username.

The list is saved to the specified file.



# CHAPTER 10

## Using Client Controls

### About Client Controls

You can:

- Import and embed client controls into a graphic.
- View and edit the properties of the client control.
- Bind the properties of the client control with attributes and element references.
- Configure scripts for client control.
- Animate client controls.
- Export a client control.
- Configure a client control with security.
- Ensure that dynamically loaded assemblies are included with the primary client control assembly when an application is deployed
- View additional client control information such as the files the client control uses and what objects and graphics are using the client control.

For information on language switching for client controls, see *Switching Languages for Graphic Elements*.

Client controls give you functionality contained in .NET controls you can use in graphics. To use this functionality, you must:

- Import the .DLL file that contains one or more client controls. The client control is imported into the Graphic Toolbox.
- Browse and embed one or more of the client controls into a new or existing graphic. The client controls appear as elements.
- View and edit the exposed client control properties.
- Bind the client control properties to attributes, graphic custom properties or tags. Do this using data binding animation.
- Configure scripts for client control. Do this using the animation.

You can then use the graphic containing the embedded client control in your HMI/SCADA software.

### Organizing Client Controls

You can organize the client controls within the Graphic Toolbox the same way as you would with Industrial Graphics. You can:

- Rename client controls.
- Move client controls in and out of Graphic Toolsets.
- Delete client controls.

For more information, see *Organizing Graphics* on page 45 and *Importing and Exporting Graphics* on page 45.

## Embedding Client Controls

You can embed an installed client control into a graphic as you would embed a graphic within another graphic.

We recommend that you not overlap client controls with other elements on the canvas. Otherwise, the client controls may not work correctly.

### To embed a client control into an Industrial graphic

1. On the **Edit** menu, click **Embed Graphic Symbol**. Your HMI's attribute/tag browser appears.
2. Browse to the location that contains the client control.
3. Select a client control from the right panel and click **OK**. The pointer changes to paste mode.
4. Click on the canvas where you want to embed the client control. The client control is placed onto the canvas.

## Viewing and Changing the Properties of Client Controls

When you embed a client control into a graphic, the native properties of the client control are imported into the Properties Editor in the **Misc** group.

Also the element container of the client control has properties such as:

- Name
- X, Y, Width, Height, AbsoluteOrigin, RelativeOrigin, and Locked
- FillColor
- TextColor and Font
- Enabled, TabOrder, TabStop, and Visible

The element container properties override the native properties of the client control.

You can view and change the properties of the control in the Properties Editor.

### To view or change the properties of a client control

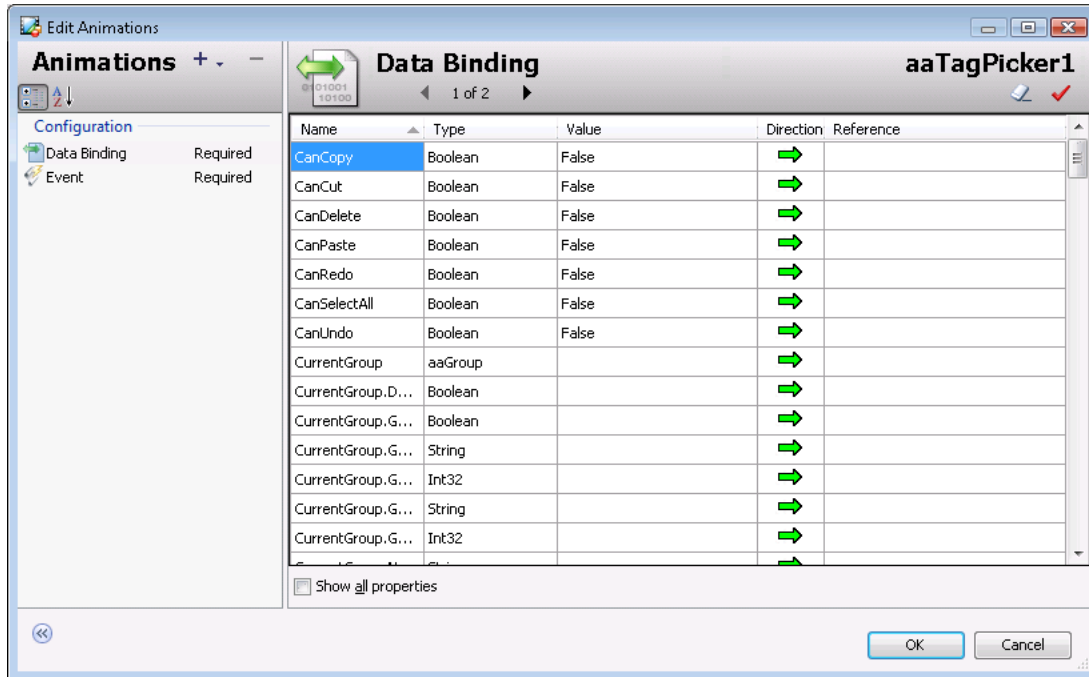
1. Select the embedded client control on the canvas.
2. In the Properties Editor, locate a:
  - Container property in the property categories **Graphic**, **Appearance**, **Fill Style**, **Text Style** or **Runtime Behavior**.
  - Native property in the **Misc** property category.
3. View or change the located property.

### To reset a client control back to its original size

- On the **Edit** menu, click **Control - Original Size**. The **AutoSize** property is set to **False**.

## Binding Client Control Properties to Attributes or Element References

You can bind the properties of an embedded client control to attributes or element references. This lets you use attributes and element references as source and consumer of data for the client control properties. You do this with the Data Binding animation.



The Data Binding table contains the following information:

- **Name** - name of the client control property
- **Type** - the .NET data type of the property
- **Value** - the default value of the client control property
- **Direction** - indicates if the property is read/write or just read-only



read/write property



read-only property



write-only property

- **Reference** - the attribute or element reference the property is bound to

**Note:** You cannot remove the **Data Binding** animation.

### To bind a client control property with an attribute or element reference

1. Double-click the embedded client control on the canvas. The **Edit Animations** dialog box appears and the **Data Binding** animation is selected by default.
2. Locate the client control property that you want to bind with an attribute or element reference.

3. Double-click the **Reference** box.
4. Do one of the following:
  - Type an attribute or element reference.
  - Browse for an attribute or element reference by clicking the **Browse** button.
5. Repeat above for any other properties you want to bind.
6. Click **OK**.

## Configuring Client Control Event Scripts

You can configure a script that is executed when a client control event occurs. You do this using the **Event** animation.

### To configure a script for a client control event

1. Double-click the embedded client control on the canvas. The **Edit Animations** dialog box appears.
2. In the animation list, click **Event**. The right panel shows the configuration.
3. In the **Event** list, select the event for which you want to execute a script. The **Parameters** list shows for the selected event:
  - **Type**: the data type of each parameter.
  - **Name**: the name of each parameter.
4. In the script area, write the event script.
5. If you want to insert an event parameter in your script, click the **Select Event Parameter** icon. Select the parameter. The parameter name is inserted into the script at the cursor position.
6. If you want to configure scripts for other, select the event from the **Event** list. The script area is cleared and you can write the script for the newly selected event.
7. When you are done, save and close.

## Animating Client Controls

Every client control has these animation types:

- Data binding animations determine which attributes and element references can read and write to the client control.
- Event animations assign scripts to individual client control.

You can add the following animations that correspond to the supported client control container properties.

- Visibility
- Fill Style
- Text Style
- Location Horizontal
- Location Vertical
- Width
- Height
- Tooltip
- Disable



If you configure these animations, the resulting behavior and appearance overrides the behavior and appearance given by the native properties of the client control.

### To add animation to embedded client controls

1. Double-click the embedded client control on the canvas. The **Edit Animations** dialog box appears.
2. Add animations as you would with any other element.

## Including Dynamically Loaded Assemblies with the Client Control

When the primary client control assembly is imported into the galaxy during an application's deployment, the system identifies all statically-linked dependent assemblies and imports them into the galaxy as well. However, if the client control contains dynamically loaded assemblies, these assemblies are not automatically loaded in the galaxy.

There are two methods for ensuring that the client control's dynamically loaded assemblies are included in the galaxy when the primary assembly is imported:

- By including the list of dynamically loaded assemblies in an XML manifest resource that is embedded in the primary assembly. The advantage of this method is that the required configuration information is packaged with the assembly, so no any other packing mechanism is required.
- By including the list of dynamically loaded assemblies in an external XML configuration file that is stored in the same directory as the primary assembly.

---

**Note:** Both methods can be used simultaneously to provide redundancy, in the event that one of the dynamically loaded assembly lists is missing a required assembly.

---

### Requirements for Both Inclusion Methods

- All dynamically loaded assemblies, as well as all non-system static dependencies of these dynamically loaded assemblies, must be stored in the same directory as the primary assembly.
- If a dynamically loaded assembly is loading another assembly dynamically, then the other assembly must be included as a dynamically loaded assembly of the primary assembly. This is a requirement because the system will not search recursively for static or dynamic dependencies.

### Sample XML for a Dynamically Loaded Assembly List

A sample list of dynamically loaded assemblies in XML format is shown below. The XML list format is the same for an embedded manifest resource or an external configuration file.

```
<?xml version="1.0" encoding="utf-8"?>
<Root>
  <DependentFiles>
    <DependentFile>
      <FileName>DyrDepAsm1.dll</FileName>
    </DependentFile>
    <DependentFile>
      <FileName>DyrDepAsm2.dll</FileName>
    </DependentFile>
    <DependentFile>
      <FileName>DyrDepAsm3.dll</FileName>
    </DependentFile>
  </DependentFiles>
</Root>
```

## XML Schema for the Dynamically Loaded Assembly List

The following XML schema is applicable for the dynamically loaded assembly XML list whether the list is provided as an embedded manifest resource or an external configuration file.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Root">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="1" maxOccurs="1" name="DependentFiles">
          <xs:complexType>
            <xs:sequence>
              <xs:element minOccurs="1" maxOccurs="unbounded"
name="DependentFile">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element minOccurs="1" maxOccurs="1"
name="FileName" type="xs:string"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:sequence>
</xs:complexType>
</xs:schema>
```

## Embedding the XML Manifest Resource in the Primary Assembly

### To embed the XML manifest resource in the primary assembly

1. In Visual Studio, add the XML list file to the client control assembly project. Any file name can be used, but the name must have the extension .aaCFG.
2. Change the **Build Action** property value to Embedded Resource.

After compilation, the XML will be available in the assembly as an embedded manifest resource file.

During the client control import operation, the system will read any embedded XML manifest resources with the extension .aaCFG. The system will then import any listed assemblies that are stored in the same location as the primary assembly.

## Including the XML Manifest Resource in an External Configuration File

### To include the XML manifest resource in an external configuration file

1. Create the XML list file using the same root name as the primary assembly but with the extension .aaCFG. For example, if the primary assembly name is MyControl.dll, then the configuration file name would be MyControl.aaCFG.
2. Store the file in the same directory as the primary assembly.

During the client control import operation, the system will look for a file that has the same root name as the primary assembly but with the extension .aaCFG and in the directory in which the primary assembly is stored. If this file is found and an embedded XML manifest resource exists, the system will consolidate the two lists to eliminate duplicate entries. The system will then import any listed assemblies that are stored in the same location as the primary assembly.

## Preventing Dynamically Loaded Assembly Import Issues

Refer to the following guidelines to prevent issues with importing the dynamically loaded assemblies.

- Make sure that the XML is valid. Invalid XML in the embedded manifest resource or the configuration file will result in the entire client control import operation for the selected primary assembly to be canceled.
- All assemblies on which the dynamically loaded assemblies are directly or indirectly dependent must be included in the same directory as the primary assembly and included in the XML list. If the system is unable to locate and load any of the direct or indirect dependencies, the entire client control import operation for the selected primary assembly will fail.
- If a dynamically loaded assembly is going to load another assembly dynamically, make sure that the other assembly is included in the XML list. If any such assemblies are not included in the primary assembly's manifest resource or configuration file, the import operation will succeed. However, these indirectly loaded assemblies will not be imported, which can result in the client control not behaving correctly during execution.



# CHAPTER 11

## Embedding Graphics within Graphics

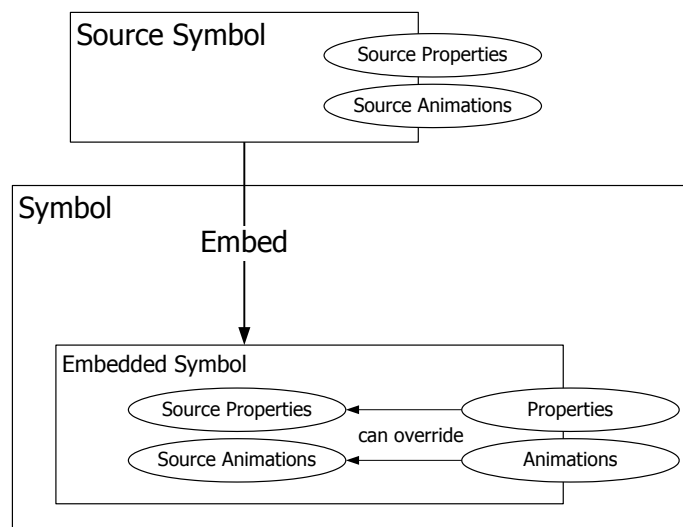
### Embedding Graphics

You can embed graphics into other graphics. This lets you split your visualization into modules and re-use already defined graphics. For example, you can create a valve graphic and embed it multiple times into a tank graphic.

When you embed a graphic into another graphic, you are creating a link to the original graphic. Any changes to the original graphic are propagated to all embedded instances.

You can:

- Embed a graphic within another graphic.
- Edit an embedded graphic.
- Restore an embedded graphic to the original size of its source graphic.
- Convert the embedded graphic to a graphic.
- Detect the source of an embedded graphic.
- Edit the source of an embedded graphic.
- Override the custom properties of the source graphic.
- Control the size propagation of an embedded graphic.
- Select an alternate or same graphic of an alternate object instance as source.
- Edit the object that contains the source graphic.
- Create a new instance of the object that contains the source graphic.



You can embed graphics from the Industrial Graphic Editor into other graphics.

When you embed a graphic, the animation links and scripts are inherited from the source graphic. You can only change the animations and scripts in the source graphic and all changes are propagated to the embedded graphic.

The embedded graphic appears with its original name appended by a number. The number is increased by one if you embed the same graphic again.

---

**Notes:**

If you embed graphics that have elements outside of the coordinates (0,0) and (2000,2000), the embedded graphic clips these elements.

The name of the embedded graphic cannot be the same as a custom property of the graphic in which it is being embedded.

The embedded graphic and the graphic in which it is being embedded cannot include elements that have the same name.

---

To embed source graphics from the Industrial Graphic Editor

1. On the Industrial Graphic Editor **Edit** menu, click **Embed Graphic Symbol**. Your HMI's attribute/tag browser appears.
2. Select a source graphic.
3. Click **OK**. The pointer appears in paste mode.
4. Click on the canvas to place the graphic.

**To embed source graphics contained in an object template**

1. On the Industrial Graphic Editor **Edit** menu, click **Embed Graphic Symbol**. Your HMI's attribute/tag browser appears.
2. Select the object template that contains the source graphic.
3. Select the graphic and click **OK**. The **Create Instance** dialog box appears.
4. Type a name for the new instance in the **New Instance Name** box and click **OK**. The new instance of the object is created and the pointer appears in paste mode.
5. Click on the canvas to place the graphic.

**To embed source graphics contained in an object instance**

1. On the Industrial Graphic Editor **Edit** menu, click **Embed Graphic Symbol**. Your HMI's attribute/tag browser appears.
2. Select the object instance that contains the source graphic.
3. Select the source graphic and click **OK**. The pointer appears in paste mode.
4. Click on the canvas to place the graphic.

## Renaming Source Graphics and Hosting Objects

Generally, if you rename source graphics or their hosting objects, embedded graphics update their references to the updated name of the renamed source graphic or hosting object.

However, if you are using relative references and you rename the contained name of the referenced object, the references to the embedded graphic are broken.

You can identify embedded graphics that may cause a problem by clicking on the embedded graphics and viewing the **SymbolReference** property from the Properties Editor. If the **SymbolReference** property contains relative references such as **me** or **myContainer**, renaming the contained name of the referenced object causes the reference to be broken.

Also, if any instance of the hosting object is checked out, when you change the contained name of the referenced object, even after you check-in the instance:

- The change is not propagated to the instance.
- Validating the object does not indicate an error.

## Editing the Embedded Graphic

After you embed a source graphic into another graphic, its animations are inherited from the source graphic. The animation of the embedded graphic is controlled by the source graphic.

The embedded graphic itself has certain animations you can configure. The animations override the animations of the source graphic for the embedded graphic. These are:

- Visibility
- Blink
- Location Horizontal
- Location Vertical
- Width
- Height
- Orientation
- Disable

Furthermore you can override the following animations if you change the `TreatAsIcon` property of the embedded graphic to `True`:

- Tooltip
- User Input
- Slider Horizontal
- Slider Vertical
- Pushbutton
- Action Scripts
- Show Symbol
- Hide Symbol

### To override the configured animations of an embedded graphic

1. Select the embedded graphic.
2. In the Properties Editor, change the value for the `TreatAsIcon` property to `True`.

## Overriding Custom Properties of the Source Graphic

You can override the value and description of a custom property of the embedded graphic if the custom property's visibility is set to `Public` in the source graphic.

You cannot add, delete, or rename any custom properties of an embedded graphic or change the data type. However, you can:

- Revert the value and description of the custom property to its default as defined in the source graphic.

- Set the visibility of the custom property. This has an effect if the graphic containing the embedded graphic is embedded into another graphic.

#### To override the value and description of a custom property

1. Select the embedded graphic on the canvas.
2. On the **Special** menu, click **Custom Properties**. The **Edit Custom Properties** dialog box appears.
3. Select the custom property you want to override with a new value or description.
4. In the **Default Value** box, type a new value.
5. In the **Description** box, type a new description.

#### To revert the value and description of a custom property

1. Select the embedded graphic on the canvas.
2. On the **Special** menu, click **Custom Properties**. The **Edit Custom Properties** dialog box appears.
3. Select the custom property you want to revert.
4. Click the Revert icon. The value and description of the selected custom property are reverted to the value and description of the same custom property in the source graphic.

## Restoring an Embedded Graphic to the Original Size of its Source Graphic

You can restore an embedded graphic to its original size as it is defined in the object or in the Industrial Graphic Editor.

#### To restore an embedded graphic to its original size

1. Select the embedded graphic that you want to restore to its original size.
2. On the **Edit** menu, point to **Embedded Symbol**, and then click **Symbol - Original Size**. The embedded graphic is restored to the original size of its source graphic.

## Converting an Embedded Graphic to a Group

You can convert an embedded graphic to a group. A converted graphic is no longer associated with its source graphic. All configuration of the embedded graphic is preserved.

If you convert an embedded graphic to a group:

- Scripts of the embedded graphic are not converted.
- You can optionally move the custom properties to the group.
- Relative references of the embedded graphic are no longer valid.

#### To convert an embedded graphic to a group

1. Select the embedded graphic that you want to convert to a group.
2. On the **Edit** menu, point to **Embedded Symbol**, and then click **Convert To Group**. The embedded graphic is converted to a group.

## Detecting the Source Graphic of an Embedded Graphic

You can view the source of an embedded graphic by using the SymbolReference property.

#### To detect the source of an embedded graphic

1. Select the embedded graphic on the canvas.



2. In the Properties Editor, view the `SymbolReference` property to see what object or environment contains the source and the name of the source graphic itself. This can be:
  - `Symbol:SymbolName`.
  - `Symbol:InstanceName.SymbolName`

## Editing the Source of an Embedded Graphic

You can edit the source of an embedded graphic by opening it in a new session of the Industrial Graphic Editor.

### To edit the source of an embedded graphic

1. Select the embedded graphic.
2. On the **Edit** menu, point to **Embedded Symbol**, and then click **Edit Symbol**. The source of the embedded graphic is opened in a new session of the Industrial Graphic Editor.
3. Edit the source graphic as needed and click **Save and Close**. The new session of the Industrial Graphic Editor is closed and the Symbol Changed icon appears in the status bar.
4. Double-click the Symbol Changed icon. The change is reflected in the embedded graphic.

If you do not accept the change, the embedded graphic is updated the next time you open it in the Industrial Graphic Editor.

## Controlling Size Propagation of Embedded Graphics

You can control the way that size changes of the source graphic are propagated to its embedded instances, which are embedded graphics. For example, a size change is:

- Resizing one of the elements in the source graphic so that the graphic boundary changes.
- Adding elements to or removing elements from the source graphic so that the graphic's boundary changes.

This feature is called dynamic size change and can be enabled or disabled.

## Setting the Anchor Point of a Source Graphic

You can set the position of the anchor point of a source graphic. The anchor point of a source graphic is by default the center point of all elements on the canvas.

You can change the position of the anchor point:

- Numerically by typing the absolute or relative anchor point position values in the Properties Editor.
- Graphically by dragging the anchor point on the canvas.

### To change the position of the anchor point numerically

1. Click on the canvas.
2. In the Properties Editor, type position values `X,Y` for:
  - **AbsoluteAnchor** property, where the position is relative to the top left corner of the canvas `0,0`.
  - **RelativeAnchor** property, where the position is relative to the center point of all elements on the canvas.

The anchor point is changed accordingly. The **AbsoluteAnchor** and **RelativeAnchor** property values are updated accordingly.

### To change the position of the anchor point graphically

1. Click on the canvas.

2. In the Properties Editor, click the **AbsoluteAnchor** or **RelativeAnchor** property label. The anchor point of the graphic is shown.
3. Drag the anchor point to the new position. The **AbsoluteAnchor** and **RelativeAnchor** property values are updated accordingly.

## Showing or Hiding the Anchor Points of Embedded Graphics

You can show or hide the anchor points of embedded graphics. An anchor point shows the current absolute anchor position of the embedded graphic on the canvas.

### To show or hide the anchor point of an embedded graphic

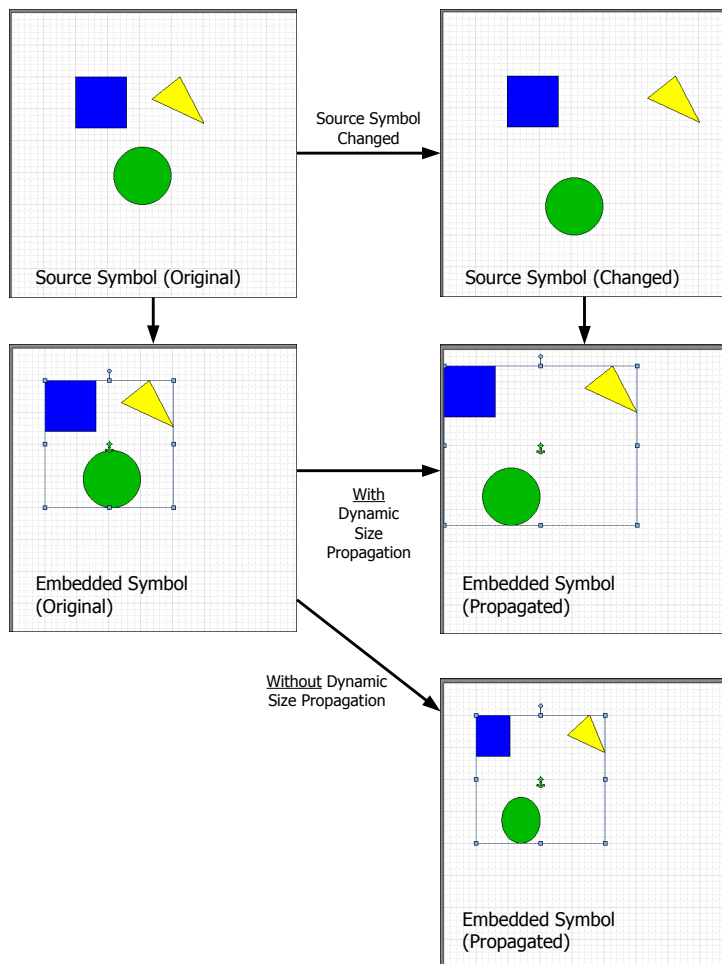
- On the toolbar, click the **Show/Hide Embedded Symbol Anchor Points** icon. The anchor of the embedded graphic appears or disappears.

## Enabling or Disabling Dynamic Size Change of Embedded Graphics

You can enable or disable the dynamic size change of embedded graphics. The anchor points of the embedded instances are not changed by any size change to the source graphic.

If the source graphic size changes and the dynamic size change is enabled, the embedded graphic size adapts accordingly. If the dynamic size change is disabled, the embedded graphic size does not change.

In both cases the anchor points of its embedded instances do not move on the canvas.



**To enable or disable dynamic size change of an embedded graphic**

1. Select the embedded graphic on the canvas.
2. On the toolbar, click the **Enable/Disable Dynamic Size Change** icon. The dynamic size change is enabled or disabled.

## Selecting Alternate Graphics and Instances

If your embedded graphic is contained in an object instance, you can:

- Use another graphic that is contained in the same object instance.
- Use the same graphic that is contained in a different object instance, in which case the animation links in the graphic are redirected.

## Selecting Alternate Graphics

You can select an alternate graphic of the same object instance to embed. The following properties are retained:

- Position and size
- Animations applied to the embedded graphic
- Override information (the TreatAsIcon property)

You can only select alternate graphics for embedded graphics contained in object instances.

**To select an alternate graphic**

1. Select the embedded graphic on the canvas.
2. On the **Edit** menu, point to **Embedded Symbol**, and then click **Select Alternate Symbol**. Your HMI's attribute/tag browser appears.
3. If available, select an alternate graphic that is contained in the same instance and click **OK**. The embedded graphic is updated with the new alternate graphic.

## Selecting Alternate Instances

You can select an alternate instance of the object that contains the same embedded graphic. When you select an alternate graphic to embed, the following properties are retained:

- Position and size
- Animations applied to the embedded graphic
- Override information (the TreatAsIcon property)

**To select an alternate instance**

1. Select the embedded graphic on the canvas.
2. On the **Edit** menu, point to **Embedded Symbol**, and then click **Select Alternate Instance**. Your HMI's attribute/tag browser appears with a list of all instances that contain the same graphic.
3. Select an instance and click **OK**. All internal references of the embedded graphic update to point at the alternate instance. The name of the embedded graphic updates to reflect that it is pointing at a different instance.

## Detecting and Editing the Containing Object Instance

You can detect and edit the object instance that contains the embedded graphic.

**To detect the object instance that contains the source graphic**

1. Select the embedded graphic.
2. In the Properties Editor, locate the `OwningObject` property. Its value contains the name of the object that contains the source graphic.

---

**Note:** You can write to this property at run time to force the embedded graphic to point to a different object in its references contained in animation links.

---

**To edit the object that contains the source graphic**

1. Select the embedded graphic.
2. On the **Edit** menu, point to **Embedded Symbol**, and then click **Edit Instance**. The object instance opens for editing in the IDE.
3. Edit the instance as needed and save your changes.

## Creating a New Instance of the Containing Object

You can create a new instance of the object that contains an embedded graphic. The following properties of the graphic are retained:

- Position and size
- Animations
- Override information (the `TreatAsIcon` property)

**To create a new instance of the object that contains an embedded graphic**

1. Select the embedded graphic.
2. On the **Edit** menu, point to **Embedded Symbol**, and then click **New Instance**. The **Create Instance** dialog box appears.
3. Type a name for the new instance in the **New Instance Name** box and click **OK**. The new instance of the object is created and the references and name of the embedded graphic are updated to point at it.

# CHAPTER 12

## Working with the Show/Hide Graphics Script Functions

### About the Show/Hide Graphic Functions

The Show/Hide Graphic script functions enable you to write Industrial Graphics scripts to display a graphic as a pop-up window and close the pop-up window.

The Show/Hide Graphics script functions are in addition to the Show/Hide Symbol animation feature, which enables you to display a graphic as a pop-up window through graphic animation. The Show/Hide Symbol animation feature remains unchanged. You can use Show/Hide Symbol animation and the Show/Hide Graphic script functions together. For more information, see *Run Time Behavior of the Show/Hide Graphic Functions* on page 252.

Like the Show/Hide Symbol animation feature, you can control the properties of the graphic through the Show Graphic feature. You can configure the script to specify:

- Which graphic will appear as the pop-up window.
- Whether the window will have a title bar.
- The initial position of the pop-up window.
- Whether the window can be resized.
- Whether the window will be modal or modeless.
- The relative position of the pop-up window.
- Passing the owning object to the graphic that you want to display.
- Values of the custom properties of the graphic.

You can use the HideSelf script function for Industrial Graphics to close the displayed graphic from within the graphic's own script.

You can use the HideGraphic() script function to close any displayed graphic given its Identity.

The ShowGraphic(), HideGraphic(), and HideSelf() functions are available in managed or published HMI applications only, if supported by your HMI.

### Configuring the Show/Hide Graphic Script Functions

You must first include a script that contains the ShowGraphic function to display a graphic as a pop-up window at run time. You can also include a script that contains the HideGraphic or HideSelf functions. The HideGraphic script function allows you to close any Industrial Graphic, displayed through the ShowGraphic script function. The HideSelf script function allows you to close the graphic, displayed by either the ShowGraphic script function or the ShowSymbol animation.

---

**Important:** The ShowGraphic function can be used in a symbol's action script, named script and pre-defined script. Although the system allows you to include it in a server script, such as Start Up, On Scan, Off Scan, Shut Down and Execute, you will not be able to execute the function at run time.

---

The HideGraphic script function can be called from any Industrial Graphic being used in the HMI application.

### To include a script that contains the Show/Hide Graphic functions within a graphic animation action script

1. Create a graphic or open an existing graphic.
2. Draw a graphic, and then double-click it to open the **Edit Animations** page.
3. Open the action script editor.
4. Click the **Display Script Function Browser** icon. The **Script Function Browser** appears.
5. In the **Graphic Client** list, click the required script function, and then click **OK**. The script is added to the graphic script editor. If you add the ShowGraphic script function, the following code snippet is added:

```
Dim graphicInfo as aaGraphic.GraphicInfo;
graphicInfo.Identity = "<Identity>";
graphicInfo.GraphicName = "<SymbolName>";
ShowGraphic( graphicInfo );
```

6. Modify the script. The Identity and GraphicName are required properties and must be specified.
  - a. You can use the **Display Graphic Browser** to set the value for the GraphicName property.
  - b. You can use the **Display Automation Object Browser** to set the OwningObject property.

For more information, see *Using the Display Graphic Browser and Display Automation Object Browser* on page 246.

For details on the scripts and samples, see *Show/Hide Graphic Script Functions Guidelines* on page 246.

## Using the Display Graphic Browser and Display Automation Object Browser

You can use the **Display Graphic Browser** to select a graphic in the Graphic Toolbox, Instances, and Relative References. You can select a graphic and insert it into the script editor.

You can use the **Display Automation Object Browser** to select an automation object and add it as an owning object. The browser displays all automation objects in the galaxy, arranged in a tree structure. The browser also displays the object containment relationship. You can select an automation object and insert it into the script editor.

---

**Note:** The automation object that you have inserted will be placed within double quotes.

---

### To select a graphic or reference name

1. On the script editor, click the **Display Graphic Browser** icon. Your HMI's attribute/tag browser appears.
2. Select the graphic, and then click **OK**. The graphic is added to the script editor.

### To select an automation object as the owning object

1. On the script editor, click the **Display Automation Object Browser** icon. Your HMI's attribute/tag browser appears.
2. Select the automation object, and then click **OK**. The automation object is added to the script editor.

## Show/Hide Graphic Script Functions Guidelines

The following sections provide script tips and guidelines, followed by scripting scenarios:

- *Using the Show/Hide Script Parameters and Properties* on page 247
- *Show/Hide Graphic Script Tips and Examples* on page 253

For information about script syntax and parameters along with basic script examples, see the *Application Server Scripting Guide*.

## Using the Show/Hide Script Parameters and Properties

The following sections provide guidelines for using the Show/Hide Graphic script parameters:

- *Using the Identity Property in the ShowGraphic() Function* on page 247
- *Height and Width Aspect Ratio* on page 247
- *Incompatible GraphicInfo Properties* on page 248

## Using the Identity Property in the ShowGraphic() Function

The Identity must be unique across the HMI application. If you want to add the HideGraphic script function, you must use the same Identity as a parameter that you have used in the ShowGraphic script. The HideSelf script function does not have any parameter.

The following table lists the various scenarios where you can use the Identity property with the ShowGraphic() function and their results in run time:

Scenario	Result in Run Time
You have executed two ShowGraphic scripts for the same graphic using the same Identity.	The first pop-up window is closed and a new one opens, displaying the same graphic.
You have executed two ShowGraphic scripts for two different graphics, but using the same Identity.	The first pop-up window displaying the first graphic is closed and a new one opens, displaying the second graphic.
You have executed two ShowGraphic scripts for the same graphic, but using different Identity properties.	Two pop-up windows are opened, displaying the same graphic.
You have executed two ShowGraphic scripts for two different graphics with different Identity properties.	Two pop-up windows are opened, displaying the two different graphics.

During configuration, the system validates only the syntax of the script. Validation of graphic and Identity existence occurs only at run time.

## Height and Width Aspect Ratio

In order to maintain aspect ratio, you can specify either the height or width of a pop-up window using the CustomizedWidthHeight property. The system calculates the unspecified property based on the graphic's aspect ratio.

If a pop-up window has a title bar, the system adjusts the size of the pop-up window so that the graphic retains its aspect ratio.

Example 1: Graphic is 100 x 100. If you specify height = 200, then the height of the content = 200 - 26 (title bar height) = 174, and width of the content = 174. The same algorithm is applied to adjust the width, based on the adjusted height.

Example 2: Graphic is 100 x 100. If you specify width = 200, then the width of the content = 200, and height of the content = 200. The same algorithm is applied to adjust the width, based on the adjusted height. The height of the container = 200 (height of the content) + 26 (height of the title bar) = 226.

If the pop-up window has a title bar, then the graphic is 100 x 100. If height = 200, then the height of the content = 200, and width of the content = 200. The same algorithm is applied to adjust the width, based on the adjusted height.

If the script contains the StretchWindowToScreenHeight property, but does not contain the Width property, the system adjusts the width of the pop-up window.

If the script contains the StretchWindowToScreenWidth property, but does not contain the Height property, the system adjusts the height of the pop-up window.

### Incompatible GraphicInfo Properties

When you call ShowGraphic with an incompatible combination of GraphicInfo properties, you will see the following warning message at run time:

```
ShowGraphic <Identity Name>. <Graphic name>.<script name> conflicting
parameters used in script: <Parameter1>, <Parameter2>
```

For example, the following incompatible properties result in a window with both Width and Height equal to 0:

```
graphicInfo.WindowRelativePosition =
aaGraphic.WindowRelativePosition.WindowXY;

graphicInfo.RelativeTo = aaGraphic.RelativeTo.Desktop;
```

In this example, a WindowRelativePosition of WindowXY is incompatible with a size RelativeTo of Desktop.

The following table shows incompatible property combinations. Shaded cells indicate incompatible GraphicInfo property combinations in addition to those specified in the Incompatible Properties column.

Window Relative Position	Size: Relative To	Incompatible Properties	Notes
Desktop	Graphic	X Y Width Height	
Desktop	Desktop	X Y Width Height	
Desktop	CustomizedWidthHeight	X Y ScalePercentage	



<b>Window Relative Position</b>	<b>Size: Relative To</b>	<b>Incompatible Properties</b>	<b>Notes</b>
Window	Graphic	X Y Width Height	
Window	Desktop	X Y Width Height	RelativeTo should be Window
Window	CustomizedWidthHeight	X Y ScalePercentage	
ClientArea	Graphic	X Y Width Height	
ClientArea	Desktop	X Y Width Height	RelativeTo should be ClientArea
ClientArea	CustomizedWidthHeight	X Y ScalePercentage	
ParentGraphic	Graphic	X Y Width Height	
ParentGraphic	Desktop	X Y Width Height	RelativeTo should be ParentGraphic

<b>Window Relative Position</b>	<b>Size: Relative To</b>	<b>Incompatible Properties</b>	<b>Notes</b>
ParentGraphic	CustomizedWidthHeight	X Y ScalePercentage	
ParentElement	Graphic	X Y Width Height	
ParentElement	Desktop	X Y Width Height	
ParentElement	CustomizedWidthHeight	X Y ScalePercentage	
Mouse	Graphic	X Y Width Height StretchWindowToScreenWidth StretchWindowToScreenHeight	
Mouse	Desktop	X Y Width Height StretchWindowToScreenWidth StretchWindowToScreenHeight	RelativeTo should be Desktop

<b>Window Relative Position</b>	<b>Size: Relative To</b>	<b>Incompatible Properties</b>	<b>Notes</b>
Mouse	CustomizedWidthHeight	X Y Width Height StretchWindowToScreenWidth StretchWindowToScreenHeight	
DesktopXY	Graphic	Width Height StretchWindowToScreenWidth StretchWindowToScreenHeight	
DesktopXY	Desktop	Width Height StretchWindowToScreenWidth StretchWindowToScreenHeight	Conflicting WindowRelative Position and RelativeTo combination
DesktopXY	CustomizedWidthHeight	ScalePercentage StretchWindowToScreenWidth StretchWindowToScreenHeight	
WindowXY	Graphic	Width Height StretchWindowToScreenWidth StretchWindowToScreenHeight	
WindowXY	Desktop	Width Height StretchWindowToScreenWidth StretchWindowToScreenHeight	Conflicting WindowRelative Position and RelativeTo combination

Window Relative Position	Size: Relative To	Incompatible Properties	Notes
WindowXY	CustomizedWidthHeight	ScalePercentage StretchWindowToScreenWidth StretchWindowToScreenHeight	
ClientAreaXY	Graphic	Width Height StretchWindowToScreenWidth StretchWindowToScreenHeight	
ClientAreaXY	Desktop	Width Height StretchWindowToScreenWidth StretchWindowToScreenHeight	Conflicting WindowRelative Position and RelativeTo combination
ClientAreaXY	CustomizedWidthHeight	ScalePercentage StretchWindowToScreenWidth StretchWindowToScreenHeight	

## Run Time Behavior of the Show/Hide Graphic Functions

The Show/Hide Graphic script functions exhibit the following behavior:

- The graphic, configured with the ShowGraphic script function, behaves like a ShowSymbol animation pop-up window, rather than a typical HMI pop-up window.
- You can configure a graphic with both the ShowAnimation and ShowGraphic scripts together. If you execute the two scripts at run time, two pop-up windows open, displaying the same or different symbols. The two pop-up windows are independent of each other.
- You can open, close, and manage the graphic from across symbols and across the entire HMI application.
- Unlike ShowSymbol animation, there is no parent/child relationship between the window that launched the graphic and the graphic launched by the ShowGraphic() script function. For more information, see *Closing a Graphic* on page 253.
- You cannot use the **Close Window** dialog box or similar dialog provided by your HMI/SCADA software at run time to close the pop-up windows displayed by the ShowGraphic script function. For more information, see *Closing a Graphic* on page 253.

- Any graphic displayed by ShowGraphic script function or ShowSymbol animation always remains in front of HMI application run-time windows, except pop-up windows that might be supported by your HMI/SCADA software. Even if you click a run-time HMI window, the window remains behind these graphics.
- If your HMI/SCADA software supports memory caching, enabling in-memory graphics caching in HMI run-time memory properties will keep ShowGraphic and ShowSymbol animation popup symbols cached in memory. The system tracks the order in which graphics are closed in order to determine their age. If a user-defined in-memory limit is exceeded, the system automatically removes the oldest popup symbols in the in-memory graphics cache except those defined in high-priority windows. If you display a graphic with the ShowGraphic script function or with ShowSymbol animation, the HMI run time will perform a memory health check if supported by your HMI/SCADA software.

## Behavior of ShowGraphic Windows with the Same Identity

ShowGraphic pop-up windows attempting to open a pop-up window with the same Identity exhibit the following behavior with the predefined scripts OnHide, OnShow, and WhileShowing:

- A ShowGraphic function within an OnShow script will be blocked if a ShowGraphic pop-up window with the same Identity is already displayed.
- A ShowGraphic function within an WhileShowing script will be blocked if a ShowGraphic pop-up window with the same Identity is already displayed.
- A ShowGraphic function within an OnHide script will be blocked if a ShowGraphic pop-up window with the same Identity is already displayed.

No error or warning messages will appear in the logger when script execution is blocked as described.

With the Graphic Cache memory option enabled, calling ShowGraphic pop-up windows with same identity name, if the graphic is modal to the modal graphic behind it, calling the ShowGraphic function cannot change this symbol to be modeless to the current modal graphic. For more information, see *Working with Modal Windows* on page 255.

## Closing a Graphic

You can close a graphic, displayed using the ShowGraphic script function, by executing the HideGraphic() or HideSelf() script functions, by clicking the **Close Window** button of the graphic pop-up window if configured, or by closing your HMI/SCADA software run time display. You cannot close the graphic by closing the HMI configuration window or the graphic that launched the graphic.

Windows opened by the ShowGraphic() script function or ShowSymbol animation are loaded dynamically and are not exposed at run time. You cannot close these windows using your HMI/SCADA software run-time **Close Window** dialog box.

## Show/Hide Graphic Script Tips and Examples

The Show/Hide Graphic script functions allow for a wide range of scripted uses. The following sections provide in-context tips and examples of script applications:

- *Using Predefined and Named Scripts* on page 254
- *Working with Modal Windows* on page 255
- *Using Hierarchical References and Containment Relationships* on page 256
- *Scripting the Owing Object* on page 257
- *Scripting Multiple Symbols* on page 262

## Using Predefined and Named Scripts

You can use the Show/Hide Graphic script functions inside container scripts. Container scripts refers to predefined scripts and named scripts. Predefined scripts include OnShow, WhileShowing, and OnHide. Named scripts include WhileTrue, WhileFalse, OnTrue, OnFalse, and DataChange. For more information, see *Predefined and Named Scripts* on page 215, *Configuring the Predefined Scripts of a Graphic* on page 219, and *Adding Named Scripts to a Graphic* on page 221.

---

**Important:** Although you can use the Show/Hide Graphic script functions inside container scripts, you cannot use ShowGraphic() in WhileTrue or periodic scripts such as WhileShowing.

---

### Container Script Scenario

The following scenario illustrates the use of Show/Hide Graphic script functions inside a container script: You want to automatically show a graphic upon closing the graphic already showing. This entails creating a ShowGraphic script for one graphic, then creating a ShowGraphic script for a second graphic inside an OnHide predefined script.

#### To execute the container script scenario

1. Create a graphic, such as a pump, called "symbol01" and another "symbol02".
2. Add a button named "Close" to symbol01 on the Industrial Graphic Editor canvas, and add an action script to the button:

```
HideSelf();
```

3. Add a button named "Show Pump" in symbol02 on the Industrial Graphic Editor canvas and add an action script to show the graphic, as in the following script example:

```
Dim graphicInfo as aaGraphic.GraphicInfo;
graphicInfo.Identity = "showpump_script001";
graphicInfo.GraphicName = "symbol01";
graphicInfo.WindowType = aaGraphic.WindowType.Modeless;
graphicInfo.WindowRelativePosition =
aaGraphic.WindowRelativePosition.Window;
graphicInfo.WindowLocation = aaGraphic.WindowLocation.Bottom;
ShowGraphic( graphicInfo);
```

4. Add an OnHide script in symbol01. In the script editor, add a ShowGraphic function for the second graphic, symbol02, as in the following script example:

```
Dim graphicInfo as aaGraphic.GraphicInfo;
graphicInfo.Identity = "showpump_script001";
graphicInfo.GraphicName = "symbol02";
graphicInfo.WindowType = aaGraphic.WindowType.Modeless;
graphicInfo.WindowRelativePosition =
aaGraphic.WindowRelativePosition.Window;
graphicInfo.WindowLocation = aaGraphic.WindowLocation.Bottom;
ShowGraphic( graphicInfo);
```

The ShowGraphic for your second graphic is now configured inside the predefined (container) script.

5. Go to run time and open the window containing the "show pump" button.

- a. Click the "show pump" button. Symbol01 displays.
- b. Click the "close button" on symbol01. Symbol02 now displays in place of symbol01.

In this scenario, you configure and demonstrate a ShowGraphic script inside a predefined script, and use it to automatically display a second graphic upon closing the first.

By extension, you can configure more graphics the same way, accessing a sequence of graphics at run time with only one button occupying your display. You can use other container scripts, such as OnShow and WhileShowing, as well as named scripts in the same manner.

## Working with Modal Windows

If you have opened a modal pop-up window using the ShowGraphic() script function, the system cannot execute the rest of the script. You must close the window to allow the system to execute the rest of the script.

If you have opened multiple modal pop-up windows, you cannot click or close the modal window stacked in the middle of the modal chain. The system will maintain the modal chain to allow pending or unprocessed scripts to process before the graphic can close. Attempts to close a window beneath a modal window are blocked.

The following examples 1 and 2 illustrate modal window behavior using the ShowGraphic() function.

The following example 3 illustrates a specific scenario of working with modal windows with the same identity name using the ShowGraphic() function while the Graphic Cache memory feature is enabled.

**Example 1:** Modeless Symbol1 (S1) opens modeless Symbol2 (S2) using a ShowGraphic() script function. Modeless graphic S2 opens modal Symbol3 (S3) using a ShowGraphic script function. In this scenario:

- S2 cannot complete its script and close (HideSelf) until S3 closes.
- You cannot close S2 using the close window button.
- You cannot close S2 using a HideGraphic (S2) script function from another window until the modal graphic S3 closes and the S2 script completes.
- You cannot close S2 using a ShowGraphic (S2) script function with the same Identity until the modal graphic S3 closes and the S2 script completes.
- You can close S1 using a HideGraphic (S1) script function from another window because the subsequent graphic S2 is modeless.

**Example 2:** Modeless Symbol1 (S1) opens modal Symbol2 (S2) using a ShowGraphic() script function. Modal graphic S2 opens modal Symbol3 (S3) using a ShowGraphic script function. In this scenario:

- S1 cannot complete its script and close (HideSelf) until S2 closes.
- S2 cannot complete its script and close (HideSelf) until S3 closes.
- You cannot close S1 or S2 using the close window buttons.
- You cannot close S2 using a HideGraphic (S2) script function from another window until the modal graphic S3 closes.
- You cannot close S1 using a HideGraphic (S1) script function from another window until the modal graphic S2 closes.
- You cannot close S2 using a ShowGraphic (S2) script function with the same Identity until the modal graphic S3 closes.
- You cannot close S1 using a ShowGraphic (S1) script function with the same Identity until the modal graphic S2 closes.

- You can close S3 using a HideGraphic script function from another window, or by using the close window button if enabled.

---

**Note:** Although you can close a graphic by opening another graphic with the same Identity, effectively replacing the original graphic, we recommend that you do not use the same Identity as a graphic opened with ShowGraphic in a modal dialog.

---

**Example 3:** With the Graphic Cache memory feature enabled, a ShowSymbol (SS) has a button to show graphic Symbol1 (S1) (modal), and another button to show graphic Symbol1 (S1) (modeless). S1 is configured to open Symbol2 (S2) with the ShowGraphic() function. In this scenario:

- Click show graphic button S1(modeless) to open pop-up S1. Click the ShowGraphic() button in S1 to open pop-up S2 with title "Graphic01".
- User can enter input into S2.
- With the pop-up open, click the show graphic button S1(modal). Pop-up S1 will open. Click the ShowGraphic() button in S1. Pop-up S2 will open with the title "Graphic01". The already open Graphic01 pop-up window will be replaced.
- With the Graphic Cache memory feature enabled, S1 cannot change from being modal to SS to being modeless to SS.
- User cannot enter input to S2.
- Alternatively, close S2 opened from modeless S1, then open modal S1, and click the ShowGraphic() button to open S2, "Graphic01".
- User can enter input to S2.

For more information about example 3, see *Behavior of ShowGraphic Windows with the Same Identity* on page 253.

## Using Hierarchical References and Containment Relationships

Placing one or more AutomationObjects within another AutomationObject results in a collection of AutomationObjects organized in a hierarchy that matches the application model, allows for better naming and manipulation, and for more precise scripting.

Using hierarchical references in scripts makes use of the fully qualified name of a contained object, including the container object's TagName.

The following table provides generic examples of using hierarchical references and containment relationships in scripts.

Without Hierarchical References	With Hierarchical References
<pre>GraphicName = "MyContainer.Contained ObjectHierarchyName.Symbol Name";</pre>	<pre>GraphicName = MyContainer.Tagname + ".ContainedObjectHierarchy Name.SymbolName"; GraphicName = me.Container + ".ContainedObjectHierarchy Name.SymbolName";</pre>
<pre>GraphicName = "MyPlaform.SymbolName";</pre>	<pre>GraphicName = MyPlaform.Tagname + ".SymbolName";</pre>
<pre>GraphicName = "MyEngine.SymbolName";</pre>	<pre>GraphicName = MyEngine.Tagname +</pre>



Without Hierarchical References	With Hierarchical References
	<code>".SymbolName";</code>
<code>GraphicName = "MyArea.SymbolName";</code>	<code>GraphicName = MyArea.Tagname + ".SymbolName";</code>

An example of a HierarchicalName is a valve object with a contained name of "Inlet" within a reactor named "Reactor1". The valve object would have "Reactor1.Inlet" as the HierarchicalName.

The valve object would also have a unique TagName distinct from its HierarchicalName, such as "Valve101".

Another example of a HierarchicalName is a level transmitter with the TagName "TIC101" placed within a container object called "Reactor1" and given the name "Level" within that container. This results in the HierarchicalName "Reactor1.Level".

## Scripting the Owing Object

The owing object in a ShowGraphic script function resolves only relative references. Any absolute reference is not affected by the owing object. The owing object is independent of the graphic definition. The relative reference is resolved by the object that hosts the script. For example, where GraphicName = "me.S1" or "Obj1.S1", and OwingObject = "Obj2", the owing object resolves only the relative reference in the graphic S1.

Consider a scenario where there are two automation object instances, "Reactor\_001" and "Reactor\_002" in a user galaxy. Both instances have four user-defined attributes, int1, int2, real1 and real2, and the graphic S1.

The graphic (Reactor\_001.S1) has the following UDA references:

- Me.int1 (relative reference)
- Me.real1 (relative reference)
- Reactor\_001.int2 (absolute reference)
- Reactor\_001.real2 (absolute reference)

If you configure Reactor\_001.S1 with the ShowGraphic script ("me.S1" or "Reactor\_001.S1", OwingObject = "Reactor\_002") and execute the script at run time, the system displays Reactor\_001.S1, though the relative reference within this graphic points to Reator\_002 object.

In such a case:

- The graphic always opens from only the host automation object instance, here "Reactor\_001".
- The GraphicName property can be set to relative reference, absolute reference or the Graphic Toolbox graphic name. If a relative reference is used in the GraphicName property, then the graphic will always open from only the host automation object instance, here "Reactor\_001". If an absolute reference or Graphic Toolbox graphic name is used in the GraphicName property such as "Reactor\_001.S1" or "S1", then the system will search for such graphic by its name.
- The relative references for Me.int1, Me.real1 is redirected to Reactor\_002.
- The absolute references for Reactor\_001.int2 and Reactor\_001.real2 come from Reactor\_001 only, and are not redirected to Reactor\_002.

Relative references (such as me.int1) used in custom property override on embedded symbols would resolve to the owing object of the embedded graphic. Whereas the relative references used in container scripts would resolve to the owing object of the graphic where the script is configured.

The following examples illustrate a couple of scenarios where you may need to use the ShowGraphic function to work with owning objects.

## Owning Object Scenario 1

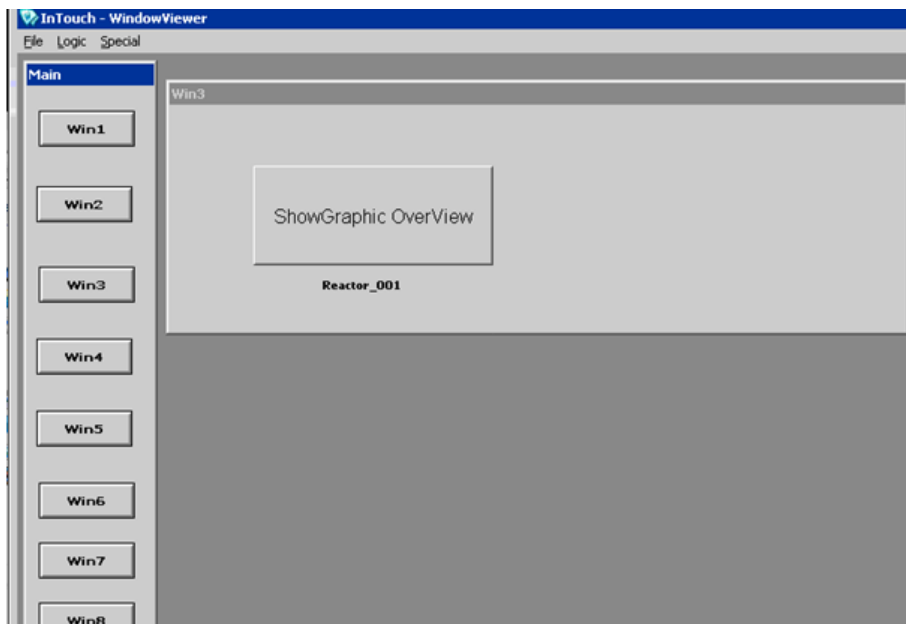
You need to monitor several similar field devices at run time, but do not want all the object windows open at the same time, as this consumes system resources and clutters the display. You can use ShowGraphic() with the OwningObject feature to rapidly switch back and forth between displays using a single interface element — a pushbutton configured with ShowGraphic() script function.

You can use the following script for the purpose:

```
Dim graphicInfo as aaGraphic.GraphicInfo;  
graphicInfo.Identity = "Overview_" + Cp1;  
graphicInfo.GraphicName = "Reactor_001.Overview";  
graphicInfo.OwningObject = cp1;  
ShowGraphic( graphicInfo );
```

Where cp1 = "Reactor\_001", "Reactor\_002", or "Reactor\_003". Object Reactor\_001, Reactor\_002, and Reactor\_003 derive from the same template \$Reactor.

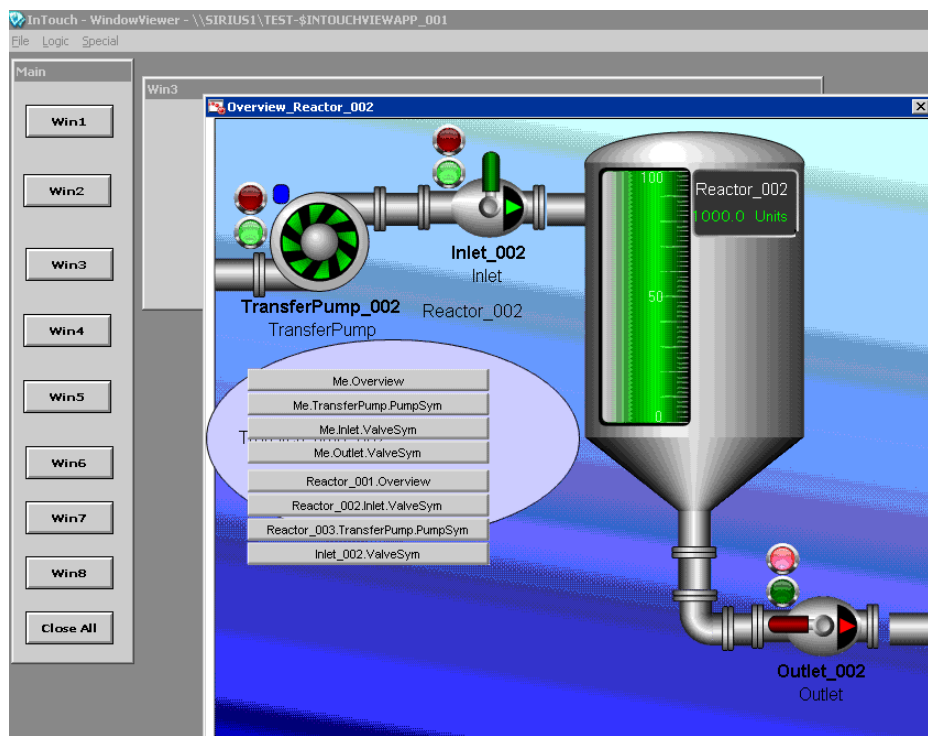
At run time, click the **ShowGraphicOverview** pushbutton to open the graphic. You can change **Reactor\_001** to **Reactor\_002** to switch between the two graphics, illustrated in the following sequence.



Displaying "Reactor\_001":



Switched to display "Reactor\_002" using the configured button:



## Owning Object Scenario 2

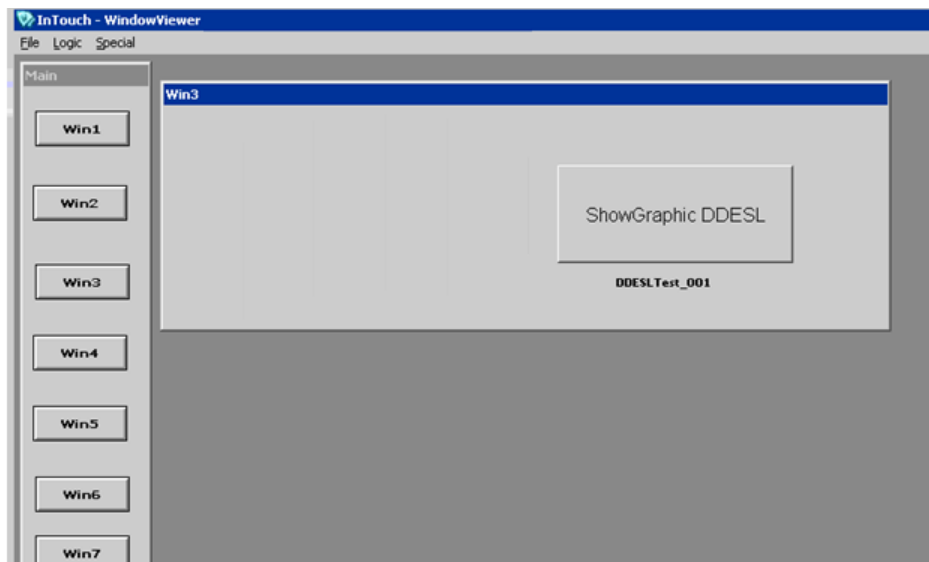
You need to monitor and maintain connection to several different data acquisition servers, but do not want to keep all server property windows open at all times. You can use ShowGraphic() with the OwningObject feature to switch back and forth between server status displays using a single interface element (pushbutton).

You can use the following script for the purpose:

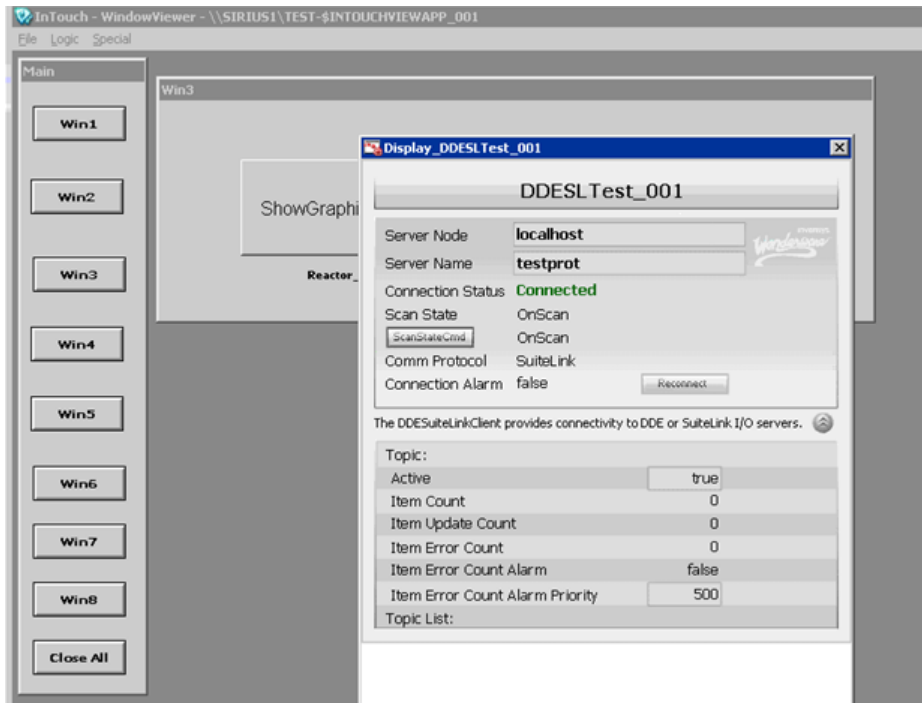
```
Dim graphicInfo as aaGraphic.GraphicInfo;  
graphicInfo.Identity = "Display_" + cp2;  
graphicInfo.GraphicName = "DDESL";  
graphicInfo.OwningObject = cp2;  
ShowGraphic( graphicInfo );
```

Where cp2 = "DDESLTest\_001", "DDESLTest\_002", and DDESL is the Graphic Toolbox graphic. DDESLTest\_001 DDESLTest\_001 are the automation object instances.

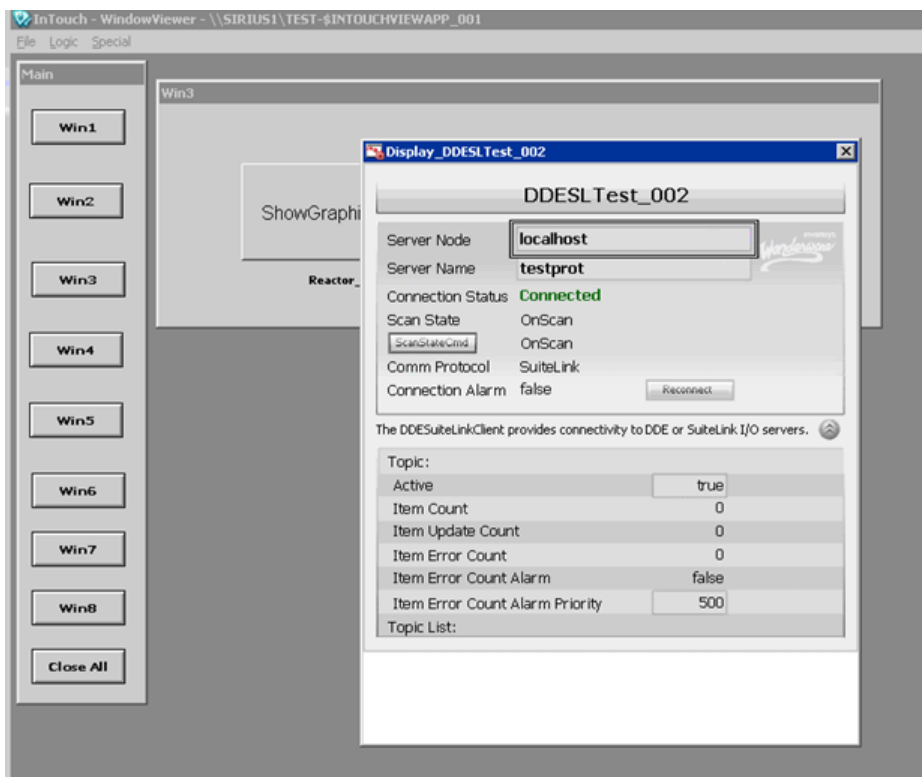
At run time, click the **ShowGraphicDDESL** pushbutton to open the graphic. You can change **DDESLTest\_001** to **DDESLTest\_002** to switch between the two graphics, illustrated in the following sequence.



Displaying server DDESL Test\_001:



Switched to display DDESL Test\_002 using the configured button:



## Assigning Custom Property Values of a Graphic

Custom properties of a graphic can be set to values when a graphic is shown by ShowGraphic() containing the CustomProperties property.

The parameters of CustomProperties are the custom property name, assigned value, and the IsConstant Boolean flag that indicates if the custom property value is a constant. Any parameter that has default value in the GraphicInfo is optional. If no input value is specified for these parameters, the default values are used at run time. Any parameter except the Enum data type can be a constant, reference, or expression.

These parameters are specified as an array of values using the CustomPropertyValuePair[] array. The array index starts at 1.

Use a script similar to the following to assign values to a symbol's custom properties. In this example, "i1" is string Identity and the graphic "S1" contains custom properties CP1 and CP2. When S1 is shown during run time, CP1 is assigned a constant value of 20 and CP2 is assigned the current value of the reference Pump.PV.Tagname.

```
Dim graphicInfo as aaGraphic.GraphicInfo;
Dim cpValues [2] as aaGraphic.CustomPropertyValuePair;
cpValues[1] = new
aaGraphic.CustomPropertyValuePair("CP1", 20, true);
cpValues[2] = new
aaGraphic.CustomPropertyValuePair("CP2",
"Pump.PV.TagName", false);
graphicInfo.Identity = "i1";
graphicInfo.GraphicName = "S1";
graphicInfo.OwningObject = "UserDefined_001";
graphicInfo.WindowTitle = "Graphic01";
graphicInfo.Resizable = false;
graphicInfo.CustomProperties=cpValues;
ShowGraphic( graphicInfo );
```

## Scripting Multiple Symbols

You can use the ShowGraphic script function to launch multiple windows from the same interface element, like a pushbutton. The following examples illustrate scenarios, where you may need to use the Show Graphic function to work with multiple symbols.

### Multiple Symbols Scenario 1

You need to open several graphics using the same interface element, such as a pushbutton.

You can use the following script for the purpose:

```
Dim graphicInfo as aaGraphic.GraphicInfo;
graphicInfo.Identity = "i1";
graphicInfo.GraphicName = "AnalogHiLo";
graphicInfo.HasTitleBar = true;
graphicInfo.WindowTitle = "Analog Meter 1";
graphicInfo.Resizable = true;
graphicInfo.WindowLocation = aaGraphic.WindowLocation.Leftside;
graphicInfo.WindowType = aaGraphic.WindowType.Modeless;
ShowGraphic( graphicInfo );

graphicInfo.Identity = "i2";
graphicInfo.GraphicName = "AnalogHiLo";
graphicInfo.HasTitleBar = true;
graphicInfo.WindowTitle = "Analog Meter 2";
graphicInfo.Resizable = true;
graphicInfo.WindowLocation = aaGraphic.WindowLocation.Center;
graphicInfo.WindowType = aaGraphic.WindowType.Modeless;
ShowGraphic( graphicInfo );

graphicInfo.Identity = "i3";
```

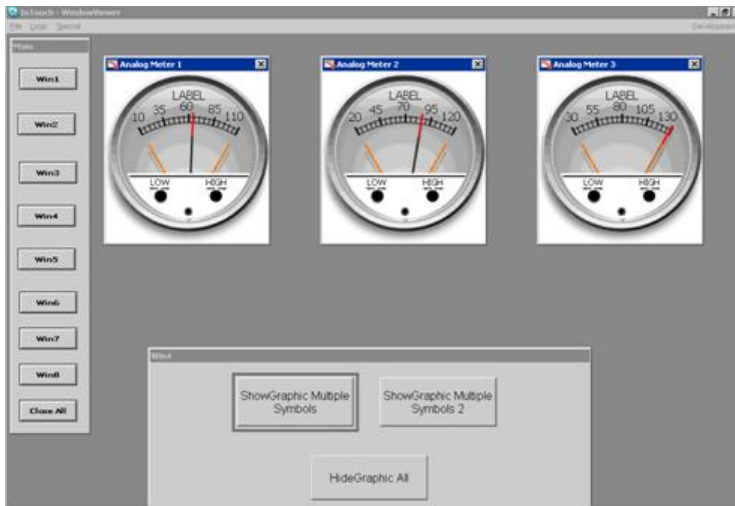
```

graphicInfo.GraphicName = "AnalogHiLo";
graphicInfo.HasTitleBar = true;
graphicInfo.WindowTitle = "Analog Meter 3";
graphicInfo.Resizable = true;
graphicInfo.WindowLocation = aaGraphic.WindowLocation.Rightside;
graphicInfo.WindowType = aaGraphic.WindowType.Modal;
ShowGraphic( graphicInfo );

```

**Note:** If you want to open multiple pop-up windows, only the last pop-up window can be modal. All other pop-up windows should be modeless. If any other pop-up window is modal, then the script will be blocked after the first modal pop-up window is opened. For more information, see *Working with Modal Windows* on page 255.

At run time, click the **ShowGraphicMultipleSymbols** pushbutton to open all the symbols:



## Multiple Symbols Scenario 2

You want to open several graphics using the same interface element, such as a pushbutton. You also want to select the graphic position and the graphic name using interface elements, like combo boxes. You can configure a combo box on the **Edit Animations** page. The combo box values can be used as index values for the window location parameter. At run time, you can dynamically select the values for the window location using this combo box.

You can use the following script for the purpose:

```

dim popup as aaGraphic.GraphicInfo;
dim MyInt as Integer;
popup.GraphicName = SelectedSymbol.Value;
IF SelectedPosition.Value == 2 THEN
    popup.Identity = "Top Left";
    popup.WindowTitle = "Top Left Corner";
ENDIF;
IF SelectedPosition.Value == 4 THEN
    popup.Identity = "TopRight";
    popup.WindowTitle = "Top Right Corner";
ENDIF;
IF SelectedPosition.Value == 9 THEN
    popup.Identity = "BottomLeft";
    popup.WindowTitle = "Bottom Left Corner";
ENDIF;
IF SelectedPosition.Value == 11 THEN

```

```
popup.Identity "BottomRight";
popup.WindowTitle = "Bottom Right Corner";
ENDIF;
popup.RelativeTo = aaGraphic.RelativeTo.CustomizedWidthHeight;
popup.width = 300;
popup.height = 300;
MyInt = StringToIntg( SelectedPosition.Value );
popup.WindowLocation = MyInt;
ShowGraphic( popup );
```

**Note:** In the script, `popup.WindowLocation = MyInt` substitutes the explicit reference with the integer index. **SelectedSymbol** is the combo box for dynamically selecting the graphic at run time and **SelectedPosition** is the combo box for dynamically selecting the window location.

At run time, click the **ShowGraphic** pushbutton to open all the symbols. You can select the graphic in the **Select a graphic** list. You can also select the location of the graphic in the **Select a position** list.

### Multiple Graphics Scenario 3

You want the Graphics in relative position with the graphic.

You can use the following script for the purpose:

```
Dim graphicInfo as aaGraphic.GraphicInfo;
graphicInfo.Identity = "i1";
graphicInfo.GraphicName = "S1";
graphicInfo.RelativeTo= aaGraphic.RelativeTo.Graphic;
ShowGraphic( graphicInfo );
```

### Multiple Graphics Scenario 4

You want the graphic in relative position with the window.

You can use the following script for the purpose:

```
Dim graphicInfo as aaGraphic.GraphicInfo;
graphicInfo.Identity = "i1";
graphicInfo.GraphicName = "S1";
graphicInfo.RelativeTo= aaGraphic.RelativeTo.Window;
ShowGraphic( graphicInfo );
```

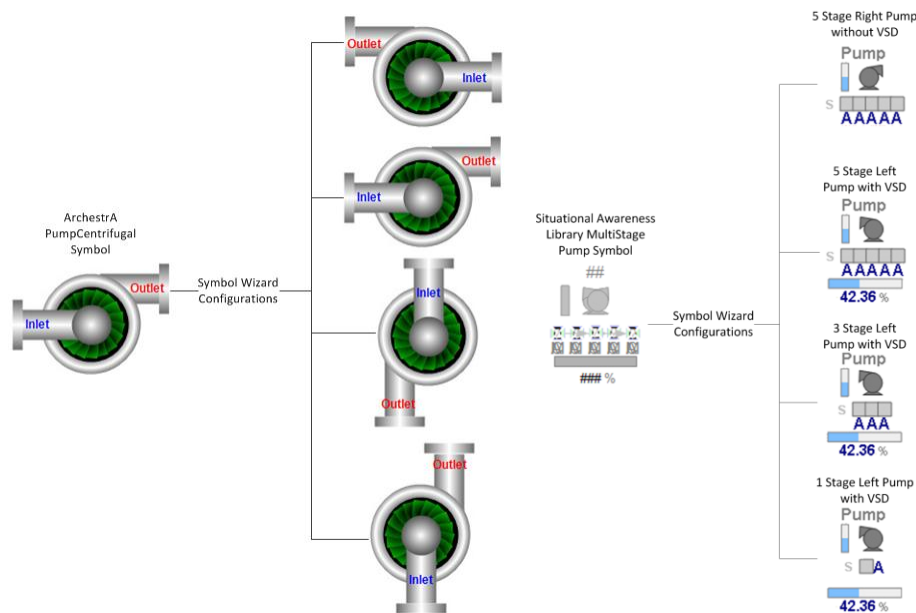


# CHAPTER 13

## Working with Symbol Wizards

### Introduction

The Industrial Graphic Editor includes the Symbol Wizard Editor, which can be used to create reusable configurable graphics called Symbol Wizards. For example, a single pump symbol can be created with the Symbol Wizard Editor that includes different visual pump configurations based on the orientation of inlet and outlet pipes. Another example of a Symbol Wizard is the Situational Awareness Library pump graphic. Situational Awareness Library graphics are designed using the Symbol Wizard Editor. However, they are protected graphics and their design cannot be changed. But, you can select Wizard Options from the Symbol Wizard Editor to select the configurations that are incorporated into each graphic's design.



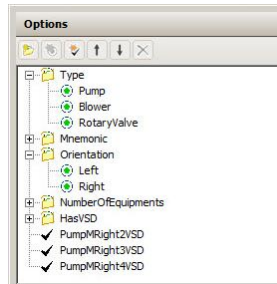
Incorporating multiple configurations in a single graphic reduces the number of graphics needed to develop an application.

### Understanding the Symbol Wizard Editor

After enabling the Symbol Wizard Editor, the Industrial Graphic Editor window updates to show Symbol Wizard Editor panes at the left of the window.

- Beneath the **Tools** pane, separate tabbed panes show the graphic elements, custom properties, and named scripts that belong to a graphic.

- The tabbed **Options** pane shows a hierarchical list of Choice Groups, Choices, and Options that define graphic configurations.



The **Options** pane includes buttons to add, delete, and reorder Choice Groups, Choices, and Options.

- The tabbed **Layers** view includes a list of defined graphic layers. Beneath each layer, separate folders contain the graphic's elements, custom properties, and named scripts associated with the layer. A graphic's elements, custom properties, and named scripts are assigned to graphic layers by dragging them to corresponding folders in the **Layers** view.

## Understanding Choice Groups and Choices

The Symbol Wizard Editor **Options** pane includes buttons to create Choice Groups, Choices, and Options.

- A Choice Group represents a unique property of a graphic and appears as the top level property node in the **Options** view.
- A Choice represents a possible value or attribute of a Choice Group property. Choices are indented beneath the associated Choice Group node in the **Options** view. Choices are mutually exclusive and only one choice can be selected from a Choice Group for a single configuration of a graphic.

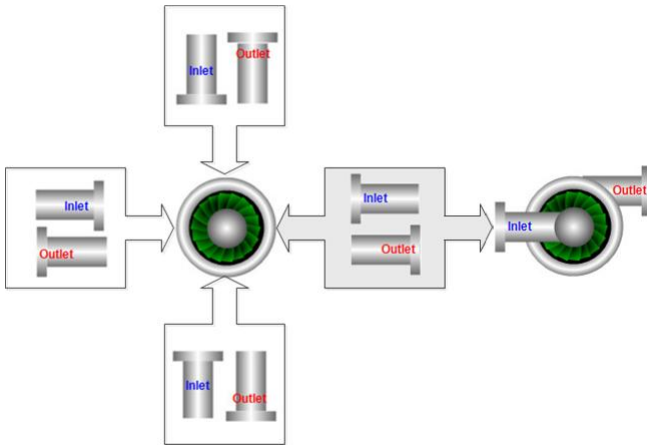
An item shown in the **Options** view list can be moved by selecting it, and then clicking the **Up** or **Down** arrow. If no Choice is specified as the default value for a Choice Group, the first Choice added to the Choice Group is always the default value.

In the example of an centrifugal pump graphic, one possible Choice Group is Orientation for the different configurations of inlet and outlet pipes. The Left, Right, Bottom, and Top choices appear as the associated Choice attributes of the Orientation Choice Group.

## Understanding Symbol Wizard Layers

Symbol Wizard layers associate graphic elements, custom properties, and named scripts to a unique graphic configuration defined by a rule. When the rule is True, the layer's graphic elements, custom properties, and named scripts are part of the Symbol Wizard's configuration.

In the example of a centrifugal pump graphic, a rule determines the orientation of the pump's inlet and outlet pipes. When the rule for the Right configuration is True, the Right layer containing the inlet and outlet pipes is part of the graphic's configuration.



The blade housing does not belong to a layer because it is common to all pump graphic configurations. Graphic elements of a graphic that do not belong to a layer appear in all graphic configurations. As a result, the pump's blade housing appears in the Left, Right, Top, and Bottom configurations of the pump by default.

Likewise, adding graphic elements, custom properties, and named scripts to a layer without a rule results in these elements appearing in all graphic configurations. Each layer must have a defined rule that specifies a True condition when the set of graphic elements, custom properties, and named scripts are part of a graphic configuration.

Associating graphic elements, named scripts, and custom properties to graphic layers involves working with the Symbol Wizard Editor **Layers** pane shown to the left of the graphic canvas.

## Defining Graphic Configuration Rules

A rule defines an expression that determines if a given choice or option and its associated graphic layer is visible or hidden based on the evaluation of the rule to true or false.

Rules can consist of a single expression or compound expressions using Boolean keywords or operator characters:

Boolean Keywords	AND, OR, NOT
Operator Characters	<ul style="list-style-type: none"> <li>• Period (.) A period concatenates a Choice Group to a Choice in a hierarchical expression.</li> <li>• Pipe ( ) A pipe evaluates to a Boolean OR.</li> <li>• Ampersand (&amp;) An ampersand evaluates to a Boolean AND.</li> <li>• Exclamation point (!) An exclamation point evaluates to a Boolean NOT.</li> <li>• Parentheses ( ) A compound expression enclosed within parentheses is evaluated before other expressions in a rule</li> </ul>

---

Any other unlisted keywords or operator characters in a rule are treated as part of the references.

- Compound expressions that include a Boolean keyword must include blank spaces around the keyword.  
*ConditionA OR ConditionB*
- Compound expressions that include an operator character that evaluates to a Boolean condition do not require blank spaces.  
*ConditionA|ConditionB*
- A property attribute must be referenced by its hierarchal Choice Group name.  
*ChoiceGroup.Choice*
- Rules cannot reference a Choice Group alone. Rule expressions must reference Choices within a Choice Group.  
*ChoiceGroup.Choice.*
- When an Option is renamed, the name change is updated in all referenced rule expressions.
- An Option or a Choice can be deleted only if no graphics are associated with their default layers.

## Examples of Graphic Configuration Rules

The following examples explain how rules specify the layers that belong to a Symbol Wizard configuration.

- *Orientation.Left&HasTach.True*  
When this rule is True, the Symbol Wizard's configuration includes a layer containing a pair of pipes with the inlet pipe oriented to the left and a tachometer.
- *Orientation.Left AND HasTach.True*  
This rule is the same as the preceding rule except that a Boolean keyword is used rather than an operator character. Note the blank spaces before and after the Boolean keyword in the rule.
- *Orientation.Right&HasTach.False*  
When this rule is True, the Symbol Wizard's configuration includes a layer containing a pair of pipes with the inlet pipe oriented to the right and a layer that does not include a tachometer.
- *(Orientation.Top&HasTach.True)|(Orientation.Bottom&HasTach.True)*  
When this rule is True, the Symbol Wizard's configuration includes two layers containing pipes with inlet pipe oriented at the top or the bottom. Both pipe layers include a tachometer. The selected option of the Orientation Wizard Option determines which pipe layer appears in the configuration.

For more practical examples of creating rules, see *Symbol Wizard Tips and Examples* on page 277.

## Designing a Symbol Wizard

The process of creating and implementing a Symbol Wizard has two workflows:

- The first workflow, referred to as a designer workflow, uses the Symbol Wizard Editor to create Symbol Wizards containing multiple configurations.
- The second workflow, referred to as a consumer workflow, embeds a Symbol Wizard and then configures it for use in an application.

## Creating Graphic Choice Groups, Choices, and Options

The following list summarizes the tasks that need to be completed in a designer workflow to create a Symbol Wizard containing multiple configurations.

- Define a graphic's Choice Groups, their Choices, and Options
- Assign rules to Choice Groups, Choices, and Options
- Associate graphic elements, custom properties, and named scripts to graphic layers
- Verify each graphic configuration with Symbol Wizard Preview

After planning the possible configurations for a graphic, Designers should know the properties and the possible attributes associated with each configuration. Designers create Choice Groups, Choices, and Options to define a graphic's properties and attributes.

---

**Important:** Situational Awareness Library symbols have predefined Choice Groups, Choices, and Options.

---

### To create graphic choice groups, choices, and options

1. In the IDE, create a copy of a graphic in the Industrial Graphic Editor for which you want to create multiple configurations.  
You can also build an entirely unique symbol from scratch and create multiple configurations of it with Symbol Wizard.
2. Check out and open the copied graphic in the Industrial Graphic Editor's canvas drawing area.
3. Click the Symbol Wizard icon shown on the Industrial Graphic Editor menu bar.  
You can also show Symbol Wizard by pressing Alt+W or selecting it as an option from the **View** menu.  
The Industrial Graphic Editor updates to show the Symbol Wizard Editor's tabbed panes at the left of the window.
4. Click the **Options** tab.
5. Click **Add Choice Group** to create a Choice Group.  
A Choice Group folder appears in the **Options** window.
6. Rename the Choice Group to assign an easily identifiable name of a property used in a graphic configuration.  
Creating a Choice Group automatically sets it to rename mode. You can also manually rename a Choice Group by right-clicking on the Choice Group and select **Rename** from the menu.
7. Repeat steps 5-6 to create as many Choice Groups as needed to define all properties of a graphics that determine its configurations.
8. Select a Choice Group folder and click **Add Choice** to add a choice beneath the select Choice Group.
9. Rename the Choice to assign an easily identifiable name of a property attribute used in a graphic configuration.
10. Repeat steps 8-9 to assign all possible Choice attributes to the Choice Groups.
11. Click **Add Option** to add an Option, which appears in the window at the same hierarchical level as Choice Groups.
12. Right-click the Option and select **Rename** to assign a name.
13. Repeat steps 11-12 to create as many Options needed to define a graphic's configurations.

## Assigning Graphic Configuration Rules

In a designer workflow, you can specify rules for a graphic's defined Choices and Options. Choice Groups should not be included in graphic configuration rules.

These rules determine the graphic elements, custom properties, and scripts that belong to a graphic configuration. For more information about rule syntax, see *Defining Graphic Configuration Rules* on page 267.

### To define graphic configuration rules

1. Show the selected graphic in the Industrial Graphic Editor with the Symbol Wizard enabled.
2. Select a Choice from the **Options** view.

The **Properties** view updates to show **Option Properties** fields. The **Name** field shows the name of the Choice you selected from the **Options** view. The **Rule** field is blank.

3. If necessary, enter a rule for the Choice.

---

**Important:** Not all Choices require rules. Specify only those rules necessary to create graphic configurations. Choices without rules are always visible.

---

4. Repeat steps 2-3 to specify rules for the remaining Choices of the graphic.
5. Select an Option from the **Options** view.

The **Name** field of the **Option Properties** view updates to show the name of the Option you selected from the **Options** view.

6. Enter a rule for the Option that defines the conditions to show or hide the Choice Groups and Choices in a configuration.
7. Enter True or False in the **Default Value** field to set the Option as part of the graphic's default configuration or not.
8. In the **Description** field, enter a description of the Option.

The description appears when the Consumer embeds the graphic and clicks on the option to configure it.

9. Repeat steps 5-8 to specify rules and optional default values for the remaining Options of the graphic.

## Updating Graphic Layers

Symbol Wizard automatically creates a set of default layers that match the hierarchical set of Choices and Options defined for a graphic. Each Choice layer has an assigned default rule containing the expression *ChoiceGroup.Choice* that defines an attribute of a graphic's property.

The default rule for an Option layer is simply the name of the Option itself. Renaming an Option automatically renames any layer rules that reference the Option.

In a designer workflow, you can update layers by adding layers to or deleting layers from the set of default layers. Also, layers can be renamed and the default rule assigned to a layer can be changed.

---

**Important:** Updating graphic layers may not be necessary if the default set of layers created for Choices and Options can create all graphic configurations.

---

If a graphic layer is renamed, it loses the link to the Option. When the Option name is updated, the layer name will not get updated with changed Option name.

### To add or delete a graphic layer

1. Show the selected graphic in the Industrial Graphic Editor with the Symbol Wizard selected.

2. Click the **Layers** tab to show the list of layers.
3. To add a layer, do the following:



- a. Click the **Add Layer** icon above the **Layers** list.

You can also add a layer by right-clicking within the layers list to show the action menu and selecting **Add**.

The new layer appears at the bottom of the list with an assigned default name.

- b. Click on the new layer to select it.
- c. Rename the new layer.

Creating a layer automatically sets it to rename mode. You can also manually rename a layer by right-clicking on the layer and select **Rename** from the menu.

4. To delete a layer, do the following:



- a. Click on the layer within the list to be deleted.
- b. Delete the layer by clicking the **Delete Layer** icon above the **Layers** list or right clicking to show the context menu and selecting **Delete**.

### To update a layer rule

1. Show the selected graphic in the Industrial Graphic Editor with the Symbol Wizard selected.
2. Click the **Layers** tab to show the list of layers.
3. Select a layer from the list whose rule needs to be updated.

The **Layer Properties** view appears and shows the current rule assigned to the selected layer Choice or Option.

4. Click within the **Rule** field to select it.
5. Update the rule.
6. Click **Save** to save the changes to the layer rule.

## Associating Configuration Elements to Graphic Layers

The basic workflow to associate graphic elements, custom properties, or named scripts to a graphic layer consists of these general steps:

1. Select a graphic layer from the **Layers** view.
2. Select items from the tabbed **Elements**, **Named Scripts**, and **Custom Properties** views to associate with the selected layer.

---

**Note:** Multiple graphic elements, custom properties, or named scripts can be selected using the Shift key to select a range of listed items or the Ctrl key to select individual items from a list.

---

3. Drag and drop the selected graphic elements, custom properties, or scripts into the **Layers** view.

Configuration elements can be associated with a graphic layer by two methods:

- **Active layer method:** Select the check box to the left of the layer name. Then, drag and drop the configuration element anywhere within the **Layers** view. The configuration element is automatically associated to the correct folder of the active layer.

- **Direct folder method:** Select a layer and expand it to show the folders for the different types of configuration elements. Then, drag and drop the configuration element directly on the folder that matches the type of configuration element.

## Associating Graphic Elements to Graphic Layers

Graphic elements show the visual properties of a graphic. In a designer workflow, you must associate graphic elements to the defined layers of a graphic.

### To associate graphic elements to graphic layers

1. Show the symbol with the Symbol Wizard Editor selected.
2. Click the **Elements** tab to show the graphic elements that belong to the graphic.
3. Click the **Layers** tab.
4. Activate a layer from the **Layers** view by selecting the check box next to the layer.  
If you prefer to add graphic elements directly to a layer's **Graphic Elements** folder with the direct folder method, simply click the layer name from the list to select it.
5. Click the box to the left of the check box to expand the layer view and show the **Graphic Elements** folder.
6. Click on the graphic element in the **Elements** view to be associated with the active graphic layer.  
You can also select the symbol element group by clicking it on the displayed graphic.
7. Using standard Windows drag and drop technique, drag the graphic element from the **Elements** view and drop it anywhere within the **Layers** view.  
If you are using the direct folder method, you must drop the graphic element directly on the selected layer's **Graphic Elements** folder.  
The selected element appears beneath the active layer's **Graphic Elements** folder.
8. Repeat steps 6-7 to select all element groups that belong to the graphic layer.  
You can also select multiple graphic elements from the **Elements** view and drop them as a set.
9. Repeat steps 4-8 to select all elements for the different layers of a graphic.  
The **Show/Hide** icon appears to the left of the **Graphic Elements** folder in the **Layers** view. Clicking the icon shows or hides the graphic elements in a layer's **Graphic Elements** folder on the graphic itself.
10. Click the **Show/Hide** icon to verify the graphic elements associated to a layer are correct for the graphic configuration.
11. Save your changes to the graphic.

## Using Shortcut Menu Commands to Edit Graphic Layer Graphic Elements

The Symbol Wizard Editor provides a set of shortcut menu commands to add graphic elements to graphic layers or remove them from layers. Using shortcut commands makes it easier to add or remove graphic elements when a complex Symbol Wizard contains many graphic elements and layers.

### Adding Graphic Elements to Active Symbol Layers

Adding graphic elements to an active layer involves selecting an active layer, selecting one or more graphic elements, and then using the **Add To Active Layers** shortcut command.

---

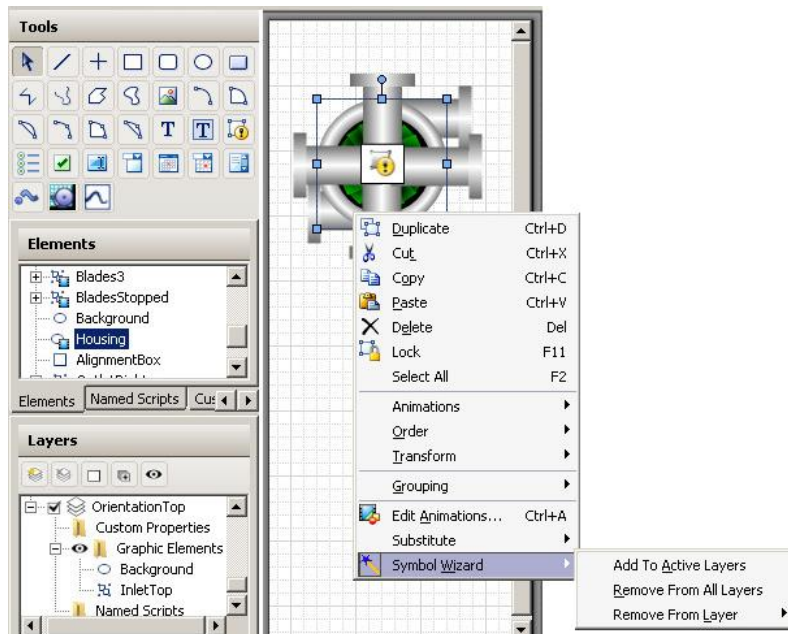
**Note:** All graphic elements should be created for all Symbol Wizard configurations before adding them to graphic layers.

---



## To add graphic elements to graphic layers

1. Show the graphic with Symbol Wizard Editor selected.
2. From the **Layers** pane, select the check box next to graphic layer to make it active.  
If you want to add a graphic element to multiple layers, select the check box next to each layer to make them active.
3. Select the graphic element to be added from the displayed Symbol Wizard.  
The graphic element can also be selected from the **Elements** pane.
4. Show the Symbol Wizard shortcut commands by right-clicking on the selected graphic element on the graphic or right-clicking on the graphic element name from the **Elements** pane.



5. Click **Add to Active Layers**.
6. Verify the graphic element has been added to the active layers.

## Removing Graphic Elements from Graphic Layers

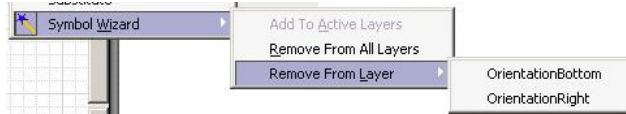
Removing graphic elements from graphic layers follows a similar sequence of steps as adding graphic elements to layers. The Symbol Wizard shortcut menu includes separate commands to remove graphic elements from all layers or only from a selected layer.

### To remove graphic elements from graphic layers

1. Show the graphic with Symbol Wizard Editor selected.
2. From the **Layers** pane, select the check box next to the graphic layer that contains a graphic element to be removed.  
Selecting a layer is not necessary if the graphic element will be removed from all layers. Also, if a layer is not selected, the **Remove From Layer** command shows a list of layers that include the selected graphic element to be removed.
3. Select the graphic element to be added from the displayed Symbol Wizard.  
The graphic element can also be selected from the **Elements** pane.
4. Show the Symbol Wizard shortcut commands by right-clicking on the selected graphic element on the graphic or right-clicking on the graphic element name from the **Elements** pane.

5. Click **Remove From All Layers** or **Remove From Layer** based on whether the graphic element should be removed from all layers or only the selected layer.

If a layer has not been selected, the **Remove From Layer** command shows a list of layers that include the graphic element selected to be removed. Click a layer from the list to remove a graphic element.



6. Verify the graphic element has been removed from the selected layers.

## Associating Custom Properties to Graphic Layers

Associating custom properties to a graphic layer uses a procedure similar to associating graphic elements. Selected custom properties are dragged and dropped on the **Custom Properties** folder to associate them to a graphic layer. You can associate custom properties to layers with the active layer or directory folder methods.

### To associate custom properties to graphic layers

1. Open the selected graphic in the Industrial Graphic Editor with the Symbol Wizard selected.
2. Click the **Custom Properties** grid to show the locally defined custom properties of the graphic. Custom properties of embedded graphics are not listed.
3. Click the **Layers** tab.
4. Select a layer from the **Layers** view to add custom properties by selecting the check box next to the layer.
5. Click the box to the left of the check box to expand the layer view and show the **Custom Properties** folder.
6. Click on a custom property in the **Custom Properties** view that belongs to the selected graphic layer.
7. Using standard Windows drag and drop technique, drag the custom property from the **Custom Properties** view and drop it on the **Custom Properties** folder.  
The selected custom property appears beneath the **Custom Properties** folder.
8. Repeat steps 6-7 to select all custom properties that belong to the graphic layer.
9. Repeat steps 4-7 to select the remaining custom properties for the different layers of a graphic.
10. Save your changes to the graphic.

## Associating Named Scripts to Graphic Layers

Associating named scripts to a graphic layer uses a similar procedure to associate graphic elements or custom properties. You can associate named scripts to layers with the active layer or directory folder methods.

### To associate named scripts to graphic layers

1. Show the selected graphic in the Industrial Graphic Editor with the Symbol Wizard selected.
2. Click the **Named Scripts** tab to show the scripts associated with the graphic.
3. Click the **Layers** tab.
4. Select a layer from the **Layers** view by selecting the check box next to the layer.

5. Click the box to the left of the check box to expand the layer view and show the **Named Scripts** folder.
6. Click on a script in the **Named Scripts** view that belongs to the selected graphic layer.
7. Using standard Windows drag and drop technique, drag the script from the **Named Scripts** view and drop it on the **Named Scripts** folder

The selected script appears beneath the **Named Scripts** folder.

8. Repeat steps 6-7 to select all scripts that belong to the graphic layer.
9. Repeat steps 4-7 to select the remaining scripts for the different layers of a graphic.
10. Save your changes to the graphic.

## Verifying Graphic Configurations

After creating the different configurations of a graphic, you can use the Symbol Wizard Preview to verify each configuration works as designed. Also, you can validate the graphic to identify any invalid references to other objects or values.

### To verify graphic configurations

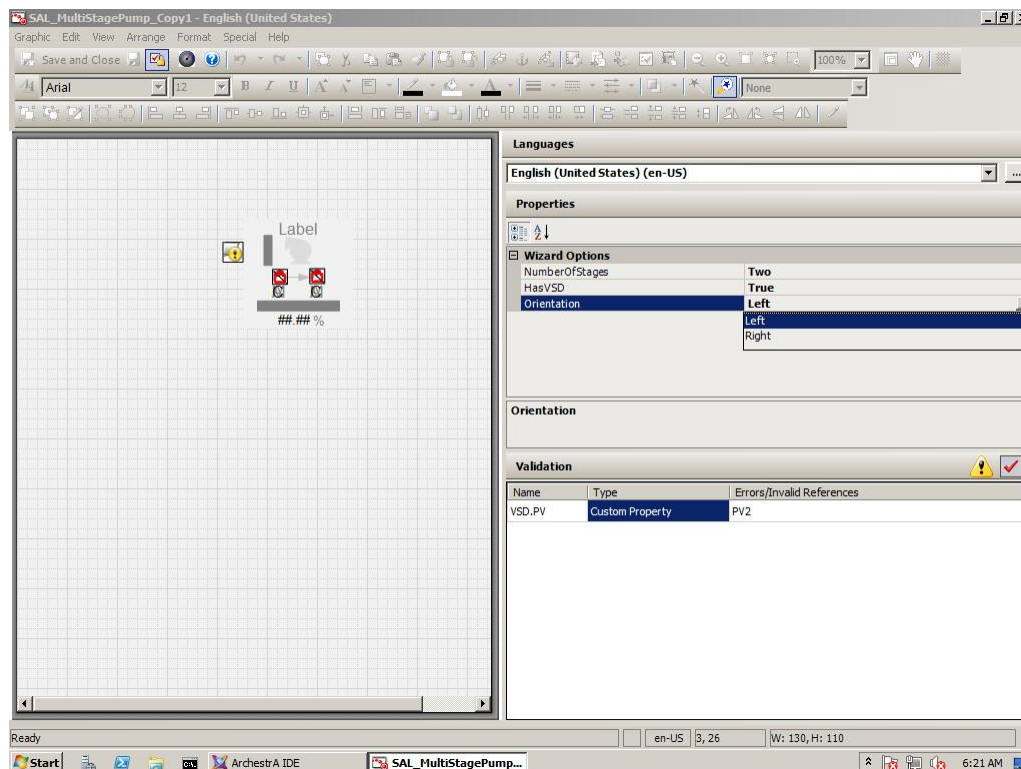
1. Open the graphic created with Symbol Wizard in the Industrial Graphic Editor.



2. Click **Symbol Wizard Preview** shown on the menu bar of the Industrial Graphic Editor.

You can also open the Symbol Wizard Preview as a **View** menu option or by pressing Alt+P.

The Industrial Graphic Editor updates to show the **Wizard Options** view with a set of drop-down lists to select different graphic property attributes and options. The default graphic configuration should be selected.



3. Select the different combinations of property values and view options from **Wizard Options** fields.

4. Verify the graphic that appears is correct for the specified configuration Choices and Option rule.



5. Click the **Validation** icon to see if the graphic contains any invalid references.

The **Validation** view lists any invalid references within the graphic that need to be corrected.

Invalid references also include references to properties or elements in hidden graphic layers.

## Using Symbol Wizards in an Application

Symbol Wizards are stored in a Galaxy library just like standard Industrial Graphics. When you select a graphic and embed it into an HMI application, the graphic's default configuration is selected.

In a consumer workflow, you can change a Symbol Wizard's configuration by changing the values assigned to the graphic's properties from the Symbol Wizard's **Wizard Options** section of the **Properties** view. After selecting a graphic configuration and changing any properties, save the graphic.

The Symbol Wizard appears as the configuration you have selected. A Symbol Wizard's configuration cannot be changed during application run time.

## Embedding Symbol Wizards

In a consumer workflow, you embed Symbol Wizards from the Industrial Graphic Editor. Embedding a Symbol Wizard is similar to embedding a standard Industrial Graphic.

A Symbol Wizard appears with its default configuration when it is embedded. The Consumer can select another configuration by changing the configuration values shown in the **Wizard Options** section of the **Properties** view.

### To embed a graphic

1. Create a new graphic or add a graphic to your HMI application.
2. Open the symbol to show the Industrial Graphic Editor.
3. On the **Edit** menu, click **Embed Graphic**.



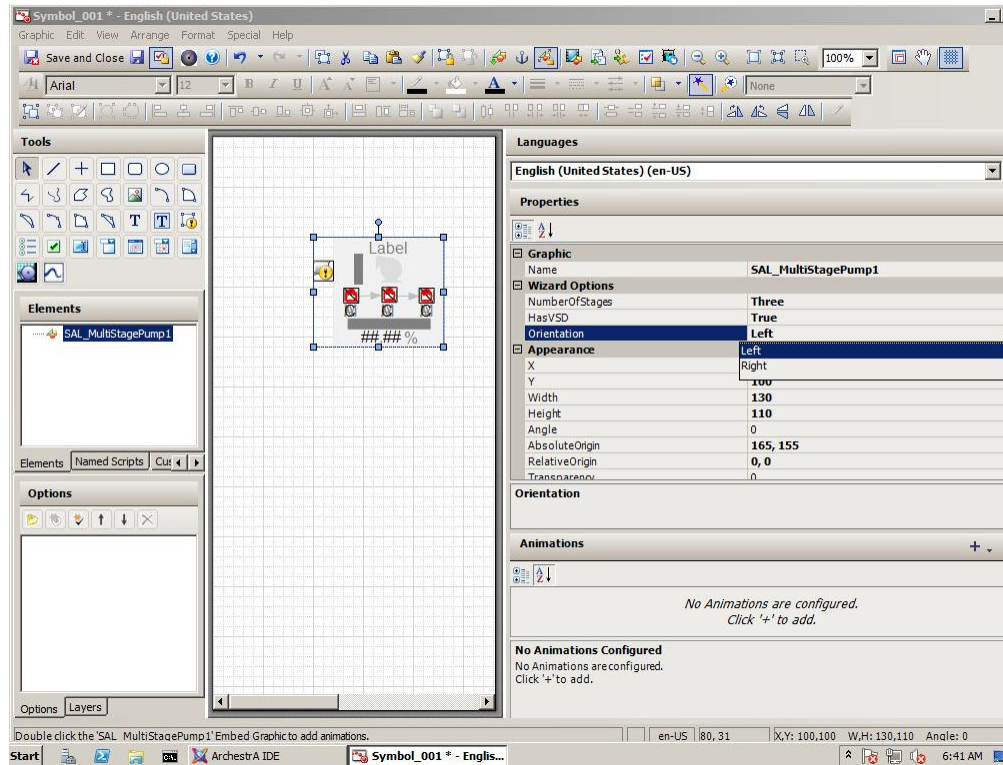
You can also click the **Embed Graphic** icon from the menu bar.

Your HMI's attribute/tag browser appears.

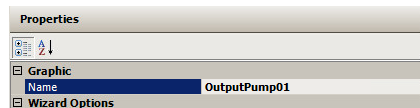
4. Locate the folder containing the Symbol Wizard.
5. Click the graphic to select it and click **OK**.
6. Position the pointer at the location where the Symbol Wizard should be placed.
7. Click once to embed the Symbol Wizard.

An embedded Symbol Wizard appears with handles on the Industrial Graphic Editor canvas to show that it is selected.

8. Select the graphic's configuration by selecting values for the various options shown in the **Wizard Options** view.



9. Rename the graphic.



10. Right-click on the graphic and select **Custom Properties** from the menu.

The **Edit Custom Properties** dialog box appears with the set of custom properties defined for the Symbol Wizard.

11. Configure the custom properties with the required references for the application.
12. Press [F10] to show the **Edit Scripts** dialog box.
13. Verify if any changes need to be made to the graphic's named scripts to run within the application.
14. Save the changes made to the graphic.

## Symbol Wizard Tips and Examples

This section describes a practical example of creating a Symbol Wizard. The example explains how to modify an centrifugal pump graphic to create a Symbol Wizard with Wizard Options that represent different orientations of inlet and outlet pumps for a tank farm application. The Symbol Wizard also includes a Wizard Option to show or hide a pump tachometer.

## Creating Visual Configurations of an Industrial Graphic

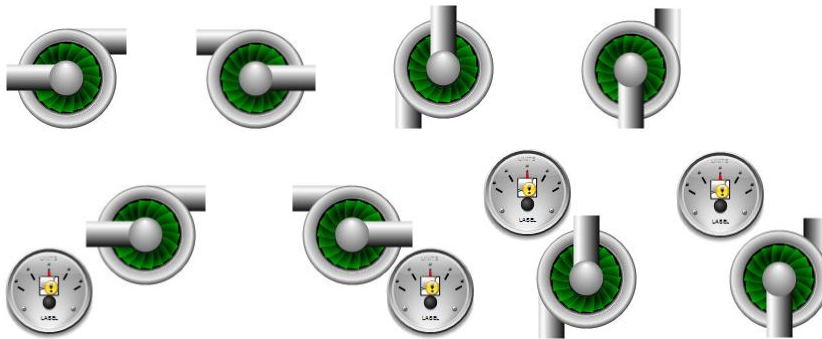
Complete the following tasks to create a Symbol Wizard:

- Plan the different configurations of a graphic and select a default configuration that represents the base Symbol Wizard.

- Identify the graphic elements needed to create each graphic configuration.
- Add graphic elements, named scripts, and custom properties for each configuration.
- Create graphic layers to group graphic elements, named scripts, and custom properties
- Specify rules to select the layers needed to create each Symbol Wizard configuration





### Planning Symbol Wizard Configurations

The first step in creating a Symbol Wizard is to identify the different configurations that should be included in a symbol. In the example of a centrifugal pump, a Symbol Wizard should represent a pump that has the inlet pipe at the left, right, top, and bottom of the pump’s central housing. Also, the Symbol Wizard should be able to show or hide a tachometer for each orientation of the centrifugal pump.







After identifying all of the different configurations of a Symbol Wizard, identify the unique properties of each configuration. The example of a centrifugal pump includes two properties: inlet pipe orientation and whether a tachometer is shown with a pump or not.

The next step is to identify the properties and associated attributes for each configuration of the symbol.

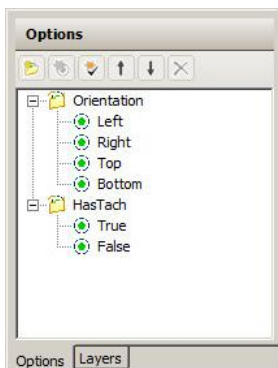
Symbol Configuration	Configuration Properties and Attributes
	Orientation=Left Has Tach=False
	Orientation=Left Has Tach=True
	Orientation=Right Has Tach=False
	Orientation=Right Has Tach=True



Symbol Configuration	Configuration Properties and Attributes
	Orientation=Top HasTach=False
	Orientation=Top HasTach=True
	Orientation=Bottom HasTach=False
	Orientation=Bottom HasTach=True

Using the Symbol Wizard Editor, create the Choice Groups and Choices needed to represent all properties and attributes of a Symbol Wizard.

In the example of a centrifugal pump, the Choice Groups are Orientation and HasTach. The Orientation Choice Group includes Left, Right, Top, and Bottom Choices, which are the possible attributes of a pump's inlet pipe. The HasTach Choice Group includes Boolean True or False Choices that indicate whether a configuration includes a tachometer or not.



Initially, the top listed Choice is the default for a Choice Group. To assign another listed Choice as the default value for the Choice Group, assign the Choice in **Default Value** field of the **Option Properties** pane.

Option Properties	
Name	Orientation
Description	
Rule	
DefaultValue	Right

If the desired base configuration of a centrifugal pump has pipes oriented to the right and includes a tachometer, then Right should be assigned as the default Orientation Choice and True assigned as the default HasTach Choice.

**Planning Tips**

- Always decide the different configurations that should be incorporated into a Symbol Wizard as the first step.
- Identify those properties that define a symbol’s configurations. These properties will be specified as the Choice Groups when building configurations with the Symbol Wizard Editor.






Identify all attributes of each property that define a symbol's configurations. These attributes will be the child Choices of the parent Choice Groups.

Select a default Symbol Wizard configuration at the planning stage to identify the symbol elements, named scripts, and custom properties that you want to include in the base configuration.




**Identify Graphic Elements**

Graphic elements are the graphics, named scripts, custom properties, and animations included with each configuration of a Symbol Wizard. The first step is to identify the graphic elements that need to be created for each Symbol Wizard configuration.

The following table shows the graphic elements needed to create a Symbol Wizard of a centrifugal pump.

Graphic Configuration	Configuration Properties and Attributes	Required Graphic Elements
	Orientation=Left HasTach=False	Graphic elements: • InletLeft • OutletRight
	Orientation=Left HasTach=True	Graphic elements: • InletLeft • OutletRight • MeterLeft
	Orientation=Right HasTach=False	Graphic elements: • InletRight • OutleftLeft
	Orientation=Right HasTach=True	Graphic elements: • InletRight • OutleftLeft • MeterRight
	Orientation=Top HasTach=False	Graphic elements: • InletTop • OutletBottom



Graphic Configuration	Configuration Properties and Attributes	Required Graphic Elements
	Orientation=Top Has Tach=True	Graphic elements: <ul style="list-style-type: none"> <li>• InletTop</li> <li>• OutletBottom</li> <li>• MeterTop</li> </ul>
	Orientation=Bottom Has Tach=False	Graphic elements: <ul style="list-style-type: none"> <li>• InletBottom</li> <li>• OutletTop</li> </ul>
	Orientation=Bottom Has Tach=True	Graphic elements: <ul style="list-style-type: none"> <li>• InletBottom</li> <li>• OutletTop</li> <li>• MeterTop</li> </ul>

### Identification Tips

- Assign short descriptive names to graphic elements. Default names are created for layers by concatenating Choice Group and Choice names. Shorter names reduce the number of Option and Layer rules that will extend beyond the borders of **Rule** field of the Symbol Wizard Editor.
- Use a standard naming convention for the graphic elements of a Symbol Wizard. Using a standard naming convention groups similar functional graphic elements together in the list shown in the **Elements** pane. This makes it easier to find graphic elements when a Symbol Wizard contains many graphic elements. Also, the names of layers appear in alphabetic order.

## Build a Visual Representation of a Symbol Wizard

The major steps to add the necessary graphic elements to a Symbol Wizard consist of the following:

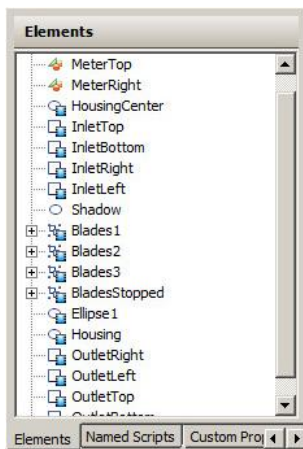
1. Check out and open an instance of a graphic from the Industrial Graphic Editor in the Industrial Graphic Editor, or create a new graphic.
2. Create the graphic elements required for each Symbol Wizard configuration by doing one of the following:
  - Embed other graphics from the Industrial Graphic Editor into the symbol.

- o Duplicate graphic elements from the symbol and edit them as necessary for each Symbol Wizard configuration.



In the example of the centrifugal pump Symbol Wizard, an meter has been embedded into centrifugal pump graphic, and then duplicated for the different configuration positions. The inlet and outlet pipes are created by duplicating the graphic pipe elements from the centrifugal pump graphic.

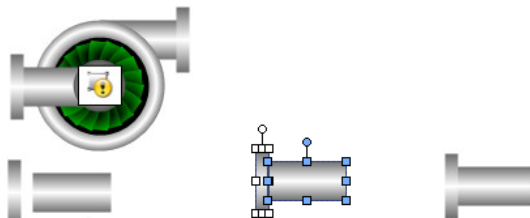
3. Rename each graphic element to easily associate it with a Symbol Wizard configuration.



4. Position the graphic elements to accurately represent the different visual representations of each configuration of a Symbol Wizard.

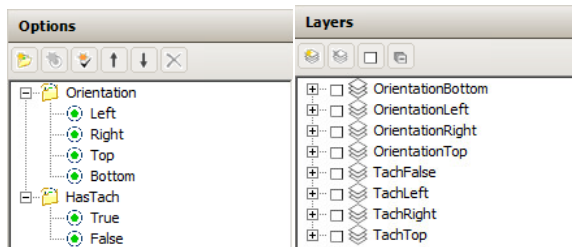
**Visualization Tips**

- If the same graphic element will be placed at different positions within a Symbol Wizard based on different configurations, create a copy of the graphic element for each position. Each graphic element can be placed into a separate layer, making it easier to specify rules to show the element at the desired position.
- If a graphic element included in a specific Symbol Wizard configuration consists of two or more elements, group the elements together. Grouping related elements makes it easier to assign graphics to Symbol Wizard layers.



## Assign Graphic Elements, Named Scripts, and Custom Properties to Graphic Layers

By default, the **Layers** pane shows a layer for each Choice Group/Choice combination listed in the **Options** pane. In the example of a centrifugal pump Symbol Wizard, there are unique layers for each orientation of the inlet pipe of the pump graphic. Inlet and outlet graphic element pairs are added to each Orientation layer.



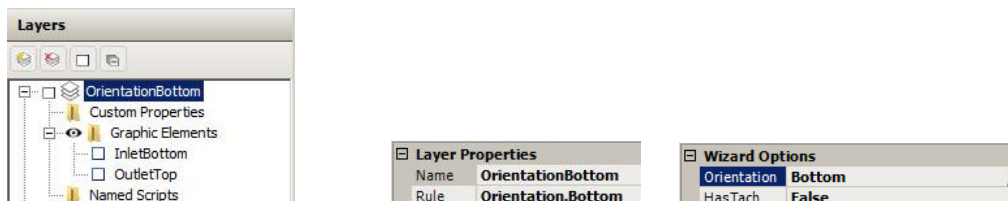
Layers need to be added for the left, right, and top positions of the tachometer when the HasTach Choice Group is True. Copies of the embedded tachometer graphic are added to the TachLeft, TachRight, and TachTop layers, which map to the different positions of the tachometer shown in the pump Symbol Wizard. The TachFalse layer does not contain any graphic elements because it selects the Symbol Wizard configurations without a tachometer.

### Layer Tips

- Use the Active Layer method to quickly drag and drop elements, scripts, and custom properties to a layer folder. Selected elements can be dropped anywhere within the **Layers** view and automatically placed in the correct folder of the active layer.
- If not created by default, create an empty layer without graphic elements for a Choice Group with Boolean True/False Choices. This makes it easier to write layer rules to hide graphic elements when a Choice Group is False.
- After adding graphic elements to a layer, toggle the **Show/Hide** icon on and off to verify the correct graphic elements have been added to the layer.
- Toggle the **Expand All/Collapse All** button above the **Layers** pane to show or hide all of the folders beneath each layer.

## Specify Rules to select Graphic Layers

A default rule is assigned to each layer based on the Choice Group Choice pair. In the example of the centrifugal pump Symbol Wizard, the Orientation Choice Group layer rules map directly to **Wizard Options** choices. Selecting an Orientation option displays the graphic elements of the selected layer in the pump's configuration.



The default layer rules to show or hide the tachometer must be modified. In the case of the Left or Right pump orientation, the layer rules must select the appropriate Orientation layer and tachometer layers. The TachLeft and TachRight layer rules include an AND statement that selects the tachometer and the Symbol Wizard's pipe orientation:

TachLeft: Orientation.Left&HasTach.True






TachRight: Orientation.Right&Has Tach.True

The layer rule to select the Top and Bottom Orientation configurations with a tachometer is more complex because the position of the tachometer is the same in both orientations. The TachTop layer rule includes separate Top and Bottom compound expressions joined with an OR statement.

(Orientation.Top&Has Tach.True)|(Orientation.Bottom&Has Tach.True)

Using Symbol Wizard Preview, verify each set of layer rules defines only a single unique Symbol Wizard configuration. Rule errors become apparent when a Symbol Wizard includes elements from other configurations, or elements are missing.

**Symbol Wizard Configuration      Wizard Options and Corresponding Active Configuration Layer Rules**

	<table border="1"> <thead> <tr> <th colspan="2">Wizard Options</th> </tr> </thead> <tbody> <tr> <td>Orientation</td> <td>Left</td> </tr> <tr> <td>HasTach</td> <td>False</td> </tr> </tbody> </table>	Wizard Options		Orientation	Left	HasTach	False
Wizard Options							
Orientation	Left						
HasTach	False						
<p>Orientation.Left&amp;Has Tach.False</p>							
	<table border="1"> <thead> <tr> <th colspan="2">Wizard Options</th> </tr> </thead> <tbody> <tr> <td>Orientation</td> <td>Right</td> </tr> <tr> <td>HasTach</td> <td>True</td> </tr> </tbody> </table>	Wizard Options		Orientation	Right	HasTach	True
Wizard Options							
Orientation	Right						
HasTach	True						
<p>Orientation.Left&amp;Has Tach.True</p>							
	<table border="1"> <thead> <tr> <th colspan="2">Wizard Options</th> </tr> </thead> <tbody> <tr> <td>Orientation</td> <td>Right</td> </tr> <tr> <td>HasTach</td> <td>False</td> </tr> </tbody> </table>	Wizard Options		Orientation	Right	HasTach	False
Wizard Options							
Orientation	Right						
HasTach	False						
<p>Orientation.Right&amp;Has Tach.False</p>							
	<table border="1"> <thead> <tr> <th colspan="2">Wizard Options</th> </tr> </thead> <tbody> <tr> <td>Orientation</td> <td>Right</td> </tr> <tr> <td>HasTach</td> <td>True</td> </tr> </tbody> </table>	Wizard Options		Orientation	Right	HasTach	True
Wizard Options							
Orientation	Right						
HasTach	True						
<p>Orientation.Right&amp;Has Tach.True</p>							
	<table border="1"> <thead> <tr> <th colspan="2">Wizard Options</th> </tr> </thead> <tbody> <tr> <td>Orientation</td> <td>Top</td> </tr> <tr> <td>HasTach</td> <td>False</td> </tr> </tbody> </table>	Wizard Options		Orientation	Top	HasTach	False
Wizard Options							
Orientation	Top						
HasTach	False						
<p>Orientation.Top&amp;Has Tach.False</p>							

## Symbol Wizard Configuration      Wizard Options and Corresponding Active Configuration Layer Rules



Wizard Options	
Orientation	Top
HasTach	True

`(Orientation.Top&HasTach.True)|(Orientation.Bottom&HasTach.True)`



Wizard Options	
Orientation	Bottom
HasTach	False

`Orientation.Bottom&HasTach.False`



Wizard Options	
Orientation	Bottom
HasTach	True

`(Orientation.Top&HasTach.True)|(Orientation.Bottom&HasTach.True)`

### Rule Tips

- Symbol Wizard rules are evaluated simultaneously. Place parentheses around compound expressions, which are evaluated before other operators outside of parentheses in a rule.
- Rules cannot reference a Choice Group alone. Always write rule expressions that reference Choices within a Choice Group in a hierarchical manner: ChoiceGroup.Choice.
- Use operator characters (&, |, !) rather than Boolean keywords (AND, OR, and NOT) to save space when writing rules. Using operator characters reduces the likelihood that a long rule will extend beyond the borders of the **Rule** field.



# APPENDIX A

## List of Element Properties

### Alphabetical List of Properties

This section shows you the properties of elements, the canvas, element groups, and embedded graphics.

Each property has a purpose, a category it belongs to, where it is used if it can be used in scripting at run time, and where to find more information on how to use it.

The first part of this section contains an alphabetical list of all properties, the second part shows a table for each category of properties.

The following table contains a list of properties used by the:

- Elements.
- Canvas.
- Element groups.
- Embedded graphics.

Asterisk (\*) marks properties that are specific to only one type of element or the canvas, a group or an embedded graphic.

Property	Purpose, category, usage and further information
AbsoluteAnchor*	<p><b>Purpose:</b> Defines the absolute anchor point of the source graphic. By default, this is the center point of all elements on the canvas but can be changed.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Canvas</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Size Propagation and Anchor Points</i> on page 34</p>
AbsoluteOrigin	<p><b>Purpose:</b> Defines an X, Y location relative to the top, left (0, 0) origin of the graphic or window.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Embedded Symbol, Group, Path, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box.</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Changing Points of Origin in the Properties Editor</i> on page 77</p>

Property	Purpose, category, usage and further information
Alignment	<p><b>Purpose:</b> Controls the location of the text relative to the bounding rectangle of the element.</p> <p><b>Category:</b> Text Style</p> <p><b>Used by:</b> Button, Text, Text Box</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Text Alignment</i> on page 101</p>
AnchorFixedTo	<p><b>Purpose:</b> Determines if the anchor point is fixed to the canvas when you resize, delete, or add elements (Absolute), or if the anchor point is recalculated relative to the element sizes and positions (Relative).</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Embedded Symbol, Canvas</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Size Propagation and Anchor Points</i> on page 34</p>
AnchorPoint*	<p><b>Purpose:</b> Defines the anchor X, Y location of the embedded graphic.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Embedded Symbol</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Size Propagation and Anchor Points</i> on page 34</p>
Angle	<p><b>Purpose:</b> Defines the current angle of rotation of the element. 0 is always the top of the element relative to the canvas. Angle is always determined relative to the top of the element and rotates in a clockwise direction.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Group, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Rotating Elements by Changing the Angle Property</i> on page 76</p>



Property	Purpose, category, usage and further information
AutoScale	<p><b>Purpose:</b> If this property is set to True then the text is stretched horizontally and vertically (larger or smaller) to fit the bounding rectangle.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Button, Text Box</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Setting Auto Scaling and Word Wrapping for a Text Box</i> on page 119</p>
ButtonStyle*	<p><b>Purpose:</b> Determines if the button appears as a standard button or as an image.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Button</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Configuring Buttons with Images</i> on page 122</p>
CalendarColumns*	<p><b>Purpose:</b> Defines the number of columns the calendar object has.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Calendar</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Number of Calendar Month Sheets</i> on page 139</p>
CalendarRows*	<p><b>Purpose:</b> Defines the number of rows the calendar object has.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Calendar</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Number of Calendar Month Sheets</i> on page 139</p>
Caption*	<p><b>Purpose:</b> Defines the text shown on the Check Box at design time and at run time when the caption property is not bound to a reference in the checkbox animation panel.</p> <p><b>Category:</b> Text Style</p> <p><b>Used by:</b> Check Box</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Caption Text of a Check Box Control</i> on page 135</p>

Property	Purpose, category, usage and further information
Checked*	<p><b>Purpose:</b> Sets or gets the value of check box. This is the initial value of the check box when the control is not connected to a reference and is overridden at run time with value of reference.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Check Box</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Setting the Default State of a Check Box Control</i> on page 135</p>
.Color1	<p><b>Purpose:</b> Color1 is a sub-property of a FillColor, UnfilledColor, LineColor or TextColor property. It is used to change the first color of the fill, unfill, line or text style if applicable.</p> <p><b>Category:</b> Depends on its source property</p> <p><b>Used by:</b> Depends on its source property</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Enabling and Disabling Elements for Run-Time Interaction</i> on page 112</p>
.Color2	<p><b>Purpose:</b> Color2 is a sub-property of a FillColor, UnfilledColor, LineColor or TextColor property. It is used to change the second color of the fill, unfill, line or text style if applicable.</p> <p><b>Category:</b> Depends on its source property</p> <p><b>Used by:</b> Depends on its source property</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Enabling and Disabling Elements for Run-Time Interaction</i> on page 112</p>
.Color3	<p><b>Purpose:</b> Color3 is a sub-property of a FillColor, UnfilledColor, LineColor or TextColor property. It is used to change the third color of the fill, unfill, line or text style if applicable.</p> <p><b>Category:</b> Depends on its source property</p> <p><b>Used by:</b> Depends on its source property</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Enabling and Disabling Elements for Run-Time Interaction</i> on page 112</p>
ControlStyle	<p><b>Purpose:</b> Defines the control style as Flat or 3D.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Radio Button Group, Check Box</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Layout of the Radio Button Group Options</i> on page 134 and <i>Setting the 3D appearance of a Check Box Control</i> on page 135</p>

Property	Purpose, category, usage and further information
Count	<p><b>Purpose:</b> Indicates how many items there are in a list.</p> <p><b>Category:</b> not available at design time</p> <p><b>Used by:</b> Radio Button Group, Combo Box, List Box</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Using Radio Button Group-Specific Properties at Run Time</i> on page 135, <i>Using Combo Box-Specific Properties at Run Time</i> on page 138 and <i>Using List Box-Specific Properties at Run Time</i> on page 143</p>
CustomFormat*	<p><b>Purpose:</b> Defines the format to be used in the DateTime Picker control for input of a date or time. <b>**inappropriateUI**</b></p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> DateTime Picker</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Configuring DateTime Picker Controls</i> on page 141</p>
CustomProperties	<p><b>Purpose:</b> The collection of CustomProperties defined by the graphic.</p> <p><b>Category:</b> Custom Properties</p> <p><b>Used by:</b> Canvas, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Using Custom Properties</i> on page 145</p>
Description*	<p><b>Purpose:</b> Contains a meaningful description of the graphic.</p> <p><b>Category:</b> Graphic</p> <p><b>Used by:</b> Canvas</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Radius of Rounded Rectangles</i> on page 117</p>
DefaultValue	<p><b>Purpose:</b> The default time value to use for the control.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Calendar, DateTime Picker</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Default Value of the Calendar Control</i> on page 141 and <i>Configuring DateTime Picker Controls</i> on page 141</p>

---

Property	Purpose, category, usage and further information
DownImage*	<p><b>Purpose:</b> Defines the image that is rendered in the button element when it is clicked or held down.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Button</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Configuring Buttons with Images</i> on page 122</p>
DropDownType*	<p><b>Purpose:</b> Defines the type of combo box: simple, drop-down or drop-down list.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Combo Box</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Type of Combo Box Control</i> on page 137</p>
DropDownWidth*	<p><b>Purpose:</b> Defines the width of the drop-down list.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Combo Box</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Width of the Drop-Down List</i> on page 137</p>
DynamicSizeChange*	<p><b>Purpose:</b> Determines if the embedded graphic propagates the size changes from the source graphic.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Embedded Symbol</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Enabling or Disabling Dynamic Size Change of Embedded Graphics</i> on page 242</p>

---

Property	Purpose, category, usage and further information
Enabled	<p><b>Purpose:</b> When set to True enables the element at run time and allows the user to interact with it. If the property is set to False the user cannot use the mouse or keyboard to interact with the element. Data changes as a result of an animation or script still execute.</p> <p><b>Category:</b> Runtime Behavior</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Enabling and Disabling Elements for Run-Time Interaction</i> on page 112</p>
End	<p><b>Purpose:</b> Defines the end of a line or H/V line as X, Y location.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Line, H/V Line</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting Start or End Points of a Line</i> on page 98</p>
EndCap	<p><b>Purpose:</b> Defines the cap used at the end of the line of an open element.</p> <p><b>Category:</b> Line Style</p> <p><b>Used by:</b> Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting Line End Shape and Size</i> on page 118</p>
FillBehavior	<p><b>Purpose:</b> Determines how the Fill (Horizontal, Vertical or Both) should be applied to the element.</p> <p><b>Category:</b> Fill Style</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting Fill Behavior</i> on page 97</p>

Property	Purpose, category, usage and further information
FillColor	<p><b>Purpose:</b> Defines the fill style used for the filled portion of the element.</p> <p><b>Category:</b> Fill Style</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, List Box, Path</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting Fill Style</i> on page 96 and <i>Changing Background Color and Text Color of Windows Common Controls</i> on page 133</p>
FillOrientation	<p><b>Purpose:</b> Determines the orientation of the fill when the element orientation is any value other than 0.</p> <p><b>Category:</b> Fill Style</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting Fill Orientation</i> on page 97</p>
FirstDayOfWeek*	<p><b>Purpose:</b> Defines the first day of the week used for the display of the columns in the calendar.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Calendar</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the First Day of the Week</i> on page 139</p>
Font	<p><b>Purpose:</b> Defines the basic text font as defined by the operating system.</p> <p><b>Category:</b> Text Style</p> <p><b>Used by:</b> Button, Text, Text Box, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Text Font</i> on page 100</p>

Property	Purpose, category, usage and further information
Format*	<p><b>Purpose:</b> Defines the format of the reference values. This is only available for array mode.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> DateTime Picker</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Configuring DateTime Picker Controls</i> on page 141</p>
HasTransparentColor*	<p><b>Purpose:</b> Indicates whether or not the image applies a transparent color. If True the image is rendered transparent wherever a color in the image matches the TransparentColor property.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Image</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Setting the Image Color Transparency</i> on page 120</p>
Height	<p><b>Purpose:</b> Defines the height of the element.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Resizing Elements by Changing Size Properties</i> on page 73</p>
HorizontalDirection	<p><b>Purpose:</b> Determines the horizontal direction of the fill for the element. Can be "Right" or "Left".</p> <p><b>Category:</b> Fill Style</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting Horizontal Fill Direction and Percentage</i> on page 97</p>

Property	Purpose, category, usage and further information
HorizontalPercentFill	<p><b>Purpose:</b> Determines the percentage of horizontal fill for the element.</p> <p><b>Category:</b> Fill Style</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Setting Horizontal Fill Direction and Percentage</i> on page 97</p>
HorizontalScrollbar	<p><b>Purpose:</b> Determines if a horizontal scroll bar appears on a list box control to allow the user to scroll the list box items horizontally.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> List Box</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Using a Horizontal Scroll Bar in a List Box Control</i> on page 143</p>
Image*	<p><b>Purpose:</b> Defines the image that is rendered in the element. Any image format supported by the application can be used.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Image</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Selecting a Different Image</i> on page 121</p>
ImageAlignment*	<p><b>Purpose:</b> Controls the location of the image relative to the bounding rectangle of the graphic. This property is only applicable when the ImageStyle is set to Normal.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Image</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Image Alignment</i> on page 120</p>
ImageStyle	<p><b>Purpose:</b> Defines how the image is rendered relative to its bounding rectangle.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Button, Image</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Image Display Mode</i> on page 119</p>



Property	Purpose, category, usage and further information
IntegralHeight	<p><b>Purpose:</b> Determines if the List Box size is an integral multiple of the Font Size so that a finite number of items fit in it without being clipped.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Combo Box, List Box</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Avoiding Clipping of Items in the Simple Combo Box Control</i> on page 138</p>
Language	<p><b>Purpose:</b> Defines the current language of the graphic.</p> <p><b>Category:</b> Runtime Behavior</p> <p><b>Used by:</b> Embedded Symbol</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> Selecting the Language for a Symbol.</p>
LanguageID	<p><b>Purpose:</b> Defines the current language ID of the graphic.</p> <p><b>Category:</b> Runtime Behavior</p> <p><b>Used by:</b> Embedded Symbol</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> Selecting the Language for a Symbol.</p>
Layout*	<p><b>Purpose:</b> Defines the way the radio buttons are arranged in the group (Horizontal or Vertical).</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Radio Button Group</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Layout of the Radio Button Group Options</i> on page 134</p>
LineColor	<p><b>Purpose:</b> Defines the color and affects of the line or border.</p> <p><b>Category:</b> Line Style</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Text Box, Path</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Line Style</i> on page 99</p>

Property	Purpose, category, usage and further information
LinePattern	<p><b>Purpose:</b> Defines the pattern of the line or border.</p> <p><b>Category:</b> Line Style</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Text Box, Path</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Line Pattern</i> on page 99</p>
LineWeight	<p><b>Purpose:</b> Determines the weight of the element's line or border. A value of 0 means that there is no line or border.</p> <p><b>Category:</b> Line Style</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Text Box, Path</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Setting the Line Weight</i> on page 99</p>
Locked	<p><b>Purpose:</b> Locks or unlocks the element's size, position, orientation and origin. Other properties that can have an affect on element size, position, orientation and origin are also locked. These are element-specific.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Locking and Unlocking Elements</i> on page 85</p>
MaxDropDownItems*	<p><b>Purpose:</b> Defines the maximum number of items the drop-down list shows.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Combo Box</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Maximum Number of Items to Appear in the Combo Box Drop-Down List</i> on page 138</p>

Property	Purpose, category, usage and further information
Multiline*	<p><b>Purpose:</b> Determines if the control shows several lines of text that automatically wrap up when reaching the right border of the control.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Edit Box</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Configuring the Text to Wrap in an Edit Box Control</i> on page 136</p>
MultiplePopupsAllowed*	<p><b>Purpose:</b> If False, ShowSymbol animations only show within a single dialog window no matter how many animations are invoked and regardless of how the animations are configured. If True, ShowSymbol animations show in separate dialog windows.</p> <p><b>Category:</b> Runtime Behavior</p> <p><b>Used by:</b> Canvas</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Radius of Rounded Rectangles</i> on page 117</p>
Name	<p><b>Purpose:</b> Gives the element a meaningful unique name.</p> <p><b>Category:</b> Graphic</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Embedded Symbol, Path</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Radius of Rounded Rectangles</i> on page 117</p>
NewIndex	<p><b>Purpose:</b> Returns the index of the last value added to the list. This is provided for migration of HMI application windows common controls.</p> <p><b>Category:</b> not available at design time</p> <p><b>Used by:</b> Combo Box, List Box</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Using Combo Box-Specific Properties at Run Time</i> on page 138 and <i>Using List Box-Specific Properties at Run Time</i> on page 143</p>

Property	Purpose, category, usage and further information
OwningObject*	<p><b>Purpose:</b> Used as the object reference to replace all "Me." references in expressions and scripts. Everywhere there is a "Me." reference this object reference is used instead. The object name can be set either using a tag or hierarchical name of an AutomationObject.</p> <p><b>Category:</b> Runtime Behavior</p> <p><b>Used by:</b> Embedded Symbol</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Detecting and Editing the Containing Object Instance</i> on page 243</p>
Radius*	<p><b>Purpose:</b> Defines the radius of the corners of the Rounded Rectangle.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Rounded Rectangle</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Setting the Radius of Rounded Rectangles</i> on page 117</p>
ReadOnly*	<p><b>Purpose:</b> Determines if the user can type data into the edit box.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Edit Box</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Configuring the Text to be Read-Only in an Edit Box Control</i> on page 136</p>
RelativeAnchor*	<p><b>Purpose:</b> Relative anchor point of the source graphic. By default, this is 0,0.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Canvas</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Size Propagation and Anchor Points</i> on page 34</p>
RelativeOrigin	<p><b>Purpose:</b> Defines the relative origin as X, Y location. The location is relative to the center point of the element (0, 0).</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Group, Path, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Changing Points of Origin in the Properties Editor</i> on page 77</p>

Property	Purpose, category, usage and further information
Scripts*	<p><b>Purpose:</b> Defines a collection of scripts configured for the graphic.</p> <p><b>Category:</b> Runtime Behavior</p> <p><b>Used by:</b> Canvas</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Adding and Maintaining Graphic Scripts</i> on page 215</p>
SelectedValue	<p><b>Purpose:</b> Reads the value of the selected item, or selects the item with that value if it exists.</p> <p><b>Category:</b> not available at design time</p> <p><b>Used by:</b> Radio Button Group, List Box</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Using Radio Button Group-Specific Properties at Run Time</i> on page 135 and <i>Using List Box-Specific Properties at Run Time</i> on page 143</p>
ShowToday*	<p><b>Purpose:</b> Determines if today's date is shown on the calendar control.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Calendar</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Showing or Hiding Today's Date on a Calendar Control</i> on page 139</p>
Smoothing*	<p><b>Purpose:</b> When False the graphics are rendered normally, when True graphics are rendered with anti-aliasing which produces a smoother appearing graphic.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Canvas</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Radius of Rounded Rectangles</i> on page 117</p>
Start	<p><b>Purpose:</b> Defines the start of a line or H/V line as X, Y location.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Line, H/V Line</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting Start or End Points of a Line</i> on page 98</p>

Property	Purpose, category, usage and further information
StartAngle	<p><b>Purpose:</b> Defines the starting angle of an Arc, Pie or Chord. 0 is always the top of the graphic relative to its orientation. A positive number is clockwise from 0 and a negative number is counter clockwise from 0. If a negative number is used to set the property it is automatically converted to a positive value.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, 2 Point Arc, 3 Point Arc</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Changing Angles of Arcs, Pies and Chords</i> on page 124</p>
StartCap	<p><b>Purpose:</b> Defines the cap used at the start of the line of an open graphic.</p> <p><b>Category:</b> Line Style</p> <p><b>Used by:</b> Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting Line End Shape and Size</i> on page 118</p>
SweepAngle	<p><b>Purpose:</b> Defines the ending angle of the Arc, Pie or Chord. This angle is always measured from the start angle.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, 2 Point Arc, 3 Point Arc</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Changing Angles of Arcs, Pies and Chords</i> on page 124</p>
SymbolReference*	<p><b>Purpose:</b> Contains the exact location that the embedded graphic is linked to. This can help the user in locating the original definition for editing purposes.</p> <p>This property is always disabled.</p> <p><b>Category:</b> Runtime Behavior</p> <p><b>Used by:</b> Embedded Symbol</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Detecting the Source Graphic of an Embedded Graphic</i> on page 240</p>

Property	Purpose, category, usage and further information
TabOrder	<p><b>Purpose:</b> Defines the tab order for the element. The tab order is only used when navigating by the keyboard. This property is valid only when the TabStop property is set to true.</p> <p><b>Category:</b> Runtime Behavior</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Editing the Tab Order of an Element</i> on page 113</p>
TabStop	<p><b>Purpose:</b> Determines if the element can be navigated to and can receive focus at run time.</p> <p><b>Category:</b> Runtime Behavior</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Editing the Tab Order of an Element</i> on page 113</p>
Tension	<p><b>Purpose:</b> Specifies how tightly the curve bends through the control points of the curve.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Closed Curve, Curve</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Changing the Tension of Curves and Closed Curves</i> on page 124</p>
Text	<p><b>Purpose:</b> Defines the unicode text that is shown by the element.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Button, Text, Text Box, Edit Box</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Setting the Displayed Text</i> on page 100</p>

Property	Purpose, category, usage and further information
TextColor	<p><b>Purpose:</b> Defines the color and affects applied to the text.</p> <p><b>Category:</b> Text Style</p> <p><b>Used by:</b> Button, Text, Text Box, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Text Color</i> on page 101 and <i>Changing Background Color and Text Color of Windows Common Controls</i> on page 133</p>
TextFormat	<p><b>Purpose:</b> Defines the formatting string that is applied to the text when it is shown.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Button, Text, Text Box</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Setting the Text Display Format</i> on page 100</p>
TitleFillColor*	<p><b>Purpose:</b> Determines the background solid color in the title bar of the calendar control.</p> <p><b>Category:</b> Fill Style</p> <p><b>Used by:</b> Calendar</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting Title Fill Color and Text Color on a Calendar Control</i> on page 140</p>
TitleTextColor*	<p><b>Purpose:</b> Determines the text solid color in the title bar of the calendar control.</p> <p><b>Category:</b> Text Style</p> <p><b>Used by:</b> Calendar</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting Title Fill Color and Text Color on a Calendar Control</i> on page 140</p>
TopIndex*	<p><b>Purpose:</b> Returns the index of the top most item in the list. This is provided for migration of HMI application windows common controls.</p> <p><b>Category:</b> not available at design time</p> <p><b>Used by:</b> List Box</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Using List Box-Specific Properties at Run Time</i> on page 143</p>



Property	Purpose, category, usage and further information
TrailingTextColor*	<p><b>Purpose:</b> Determines the text solid color of the text for the trailing days. The trailing days are days outside the current month.</p> <p><b>Category:</b> Text Style</p> <p><b>Used by:</b> Calendar</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Text Color for Trailing Dates in a Calendar Control</i> on page 140</p>
Transparency	<p><b>Purpose:</b> Defines the transparency of the element. A value of 0 means fully opaque and a value of 100 means fully transparent.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Group, Path, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Setting the Transparency Level of an Element</i> on page 111</p>
.Transparency	<p><b>Purpose:</b> Transparency is a sub-property of a FillColor, UnfilledColor, LineColor or TextColor property. It is used to change the transparency of the fill, unfill, line or text style if applicable. The transparency acts in addition to the transparency of the element.</p> <p><b>Category:</b> Depends on its source property</p> <p><b>Used by:</b> Depends on its source property</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Enabling and Disabling Elements for Run-Time Interaction</i> on page 112</p>
TransparentColor*	<p><b>Purpose:</b> Defines the RGB color value that is used as the transparent color.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Image</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Image Color Transparency</i> on page 120</p>

Property	Purpose, category, usage and further information
TreatAsIcon	<p><b>Purpose:</b> If this property is set to False, the animations defined on the graphics within the group or embedded graphic take precedence over an animation defined on the group or embedded graphic. If there are no animations or the user clicked on an area of the group or embedded graphic that does not have an animation, then the group or embedded graphic animation executes.</p> <p>If the property is set to True, only the animation on the group or embedded graphic is executed. The interactive animations within the group or embedded graphic never execute.</p> <p><b>Category:</b> Runtime Behavior</p> <p><b>Used by:</b> Group, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Editing the Embedded Graphic</i> on page 239</p>
UnFilledColor	<p><b>Purpose:</b> Determines the element's unfilled area appearance.</p> <p><b>Category:</b> Fill Style</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting Unfilled Style</i> on page 97</p>
UpImage*	<p><b>Purpose:</b> Defines the image that is used in the button element when it is un-clicked or released.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Button</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Configuring Buttons with Images</i> on page 122</p>
Value	<p><b>Purpose:</b> Reads the value of the selected item, or selects the item with that value if it exists. Its data type depends on the control.</p> <p><b>Category:</b> not available at design time</p> <p><b>Used by:</b> Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Reading and Writing the Selected Value at Run Time</i> on page 133</p>

Property	Purpose, category, usage and further information
VerticalDirection	<p><b>Purpose:</b> Defines the vertical direction of the fill. Can be "Top" or "Bottom".</p> <p><b>Category:</b> Fill Style</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting Vertical Fill Direction and Percentage</i> on page 98</p>
VerticalPercentFill	<p><b>Purpose:</b> Determines the percentage of vertical fill for the element.</p> <p><b>Category:</b> Fill Style</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Setting Vertical Fill Direction and Percentage</i> on page 98</p>
Visible	<p><b>Purpose:</b> Determines the visibility of the element. This property is configured at design time and used only at runtime. At design time all elements are visible irrespective of this setting.</p> <p><b>Category:</b> Runtime Behavior</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Changing the Visibility of Elements</i> on page 113</p>
Width	<p><b>Purpose:</b> Defines the width of the element.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Resizing Elements by Changing Size Properties</i> on page 73</p>

Property	Purpose, category, usage and further information
WordWrap	<p><b>Purpose:</b> When set to True, the text in the button or text box is formatted to fit as much text on a single line within the horizontal bounding area of the element and then continued to the next line. This continues as long as there is vertical space.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Button, Text Box</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Wrapping Text in Buttons</i> on page 122</p>
X	<p><b>Purpose:</b> Defines the left position of the element.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Moving Elements</i> on page 67</p>
Y	<p><b>Purpose:</b> Defines the top position of the element.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Moving Elements</i> on page 67</p>

## List by Functional Area

Each property of the elements, the canvas, element groups and embedded objects belongs to one of the following property categories:

- Graphic
- Appearance
- Fill Style
- Line Style
- Text Style

- Runtime Behavior
- Custom Properties

## Graphic Category Properties

The following table contains a list of properties in the Graphic property category used by the:

- Elements.
- Canvas.
- Element groups.
- Embedded graphics.

It shows their purpose, where they are used and where to find more information on how to use them.

An asterisk (\*) identifies properties that are specific to only one type of element or the canvas, a group or an embedded graphic.

Property	Purpose, category, usage and further information
Description*	<p><b>Purpose:</b> Contains a meaningful description of the graphic.</p> <p><b>Category:</b> Graphic</p> <p><b>Used by:</b> Canvas</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Radius of Rounded Rectangles</i> on page 117</p>
Name	<p><b>Purpose:</b> Gives the element a meaningful unique name.</p> <p><b>Category:</b> Graphic</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Embedded Symbol, Path</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Radius of Rounded Rectangles</i> on page 117</p>

## Appearance Category Properties

The following table contains a list of properties in the Appearance property category used by the:

- Elements.
- Canvas.
- Element groups.
- Embedded graphics.

It shows their purpose, where they are used and where to find more information on how to use them.

Asterisk (\*) marks properties that are specific to only one type of element or the canvas, a group or an embedded graphic.

Property	Purpose, category, usage and further information
AbsoluteAnchor*	<p><b>Purpose:</b> Defines the absolute anchor point of the source graphic. By default, this is the center point of all elements on the canvas but can be changed.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Canvas</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Size Propagation and Anchor Points</i> on page 34</p>
AnchorFixedTo	<p><b>Purpose:</b> Determines if the anchor point is fixed to the canvas when you resize, delete, or add elements (Absolute), or if the anchor point is recalculated relative to the element sizes and positions (Relative).</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Embedded Symbol, Canvas</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Size Propagation and Anchor Points</i> on page 34</p>
AbsoluteOrigin	<p><b>Purpose:</b> Defines an X, Y location relative to the top, left (0, 0) origin of the graphic or window.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Embedded Symbol, Group, Path, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box.</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Changing Points of Origin in the Properties Editor</i> on page 77</p>
AnchorPoint*	<p><b>Purpose:</b> Defines the anchor X, Y location of the embedded graphic.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Embedded Symbol</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Size Propagation and Anchor Points</i> on page 34</p>

Property	Purpose, category, usage and further information
Angle	<p><b>Purpose:</b> Defines the current angle of rotation of the element. 0 is always the top of the element relative to the canvas. Angle is always determined relative to the top of the element and rotates in a clockwise direction.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Group, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Rotating Elements by Changing the Angle Property</i> on page 76</p>
AutoScale	<p><b>Purpose:</b> If this property is set to True then the text is stretched horizontally and vertically (larger or smaller) to fit the bounding rectangle.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Button, Text Box</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Setting Auto Scaling and Word Wrapping for a Text Box</i> on page 119</p>
ButtonStyle*	<p><b>Purpose:</b> Determines if the button appears as a standard button or as an image.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Button</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Configuring Buttons with Images</i> on page 122</p>
CalendarColumns*	<p><b>Purpose:</b> Defines the number of columns the calendar object has.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Calendar</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Number of Calendar Month Sheets</i> on page 139</p>
CalendarRows*	<p><b>Purpose:</b> Defines the number of rows the calendar object has.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Calendar</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Number of Calendar Month Sheets</i> on page 139</p>

Property	Purpose, category, usage and further information
Checked*	<p><b>Purpose:</b> Sets or gets the value of check box. This is the initial value of the check box when the control is not connected to a reference and is overridden at run time with value of reference.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Check Box</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Setting the Default State of a Check Box Control</i> on page 135</p>
ControlStyle	<p><b>Purpose:</b> Defines the control style as Flat or 3D.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Radio Button Group, Check Box</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Layout of the Radio Button Group Options</i> on page 134 and <i>Setting the 3D appearance of a Check Box Control</i> on page 135</p>
CustomFormat*	<p><b>Purpose:</b> Defines the format to be used in the DateTime Picker control for input of a date or time.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> DateTime Picker</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Configuring DateTime Picker Controls</i> on page 141</p>
DefaultValue	<p><b>Purpose:</b> The default time value to use for the control.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Calendar, DateTime Picker</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Default Value of the Calendar Control</i> on page 141 and <i>Configuring DateTime Picker Controls</i> on page 141</p>
DownImage*	<p><b>Purpose:</b> Defines the image that is rendered in the button element when it is clicked or held down.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Button</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Configuring Buttons with Images</i> on page 122</p>



Property	Purpose, category, usage and further information
DropDownType*	<p><b>Purpose:</b> Defines the type of combo box: simple, drop-down or drop-down list.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Combo Box</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Type of Combo Box Control</i> on page 137</p>
DropDownWidth*	<p><b>Purpose:</b> Defines the width of the drop-down list.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Combo Box</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Width of the Drop-Down List</i> on page 137</p>
DynamicSizeChange*	<p><b>Purpose:</b> Determines if the embedded graphic propagates the size changes from the source graphic.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Embedded Symbol</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Enabling or Disabling Dynamic Size Change of Embedded Graphics</i> on page 242</p>
End	<p><b>Purpose:</b> Defines the end of a line or H/V line as X, Y location.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Line, H/V Line</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting Start or End Points of a Line</i> on page 98</p>
FirstDayOfWeek*	<p><b>Purpose:</b> Defines the first day of the week used for the display of the columns in the calendar.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Calendar</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the First Day of the Week</i> on page 139</p>

Property	Purpose, category, usage and further information
Format*	<p><b>Purpose:</b> Defines the format of the reference values. This is only available for array mode.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> DateTime Picker</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Configuring DateTime Picker Controls</i> on page 141</p>
HasTransparentColor*	<p><b>Purpose:</b> Indicates whether or not the image applies a transparent color. If True the image is rendered transparent wherever a color in the image matches the TransparentColor property.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Image</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Setting the Image Color Transparency</i> on page 120</p>
Height	<p><b>Purpose:</b> Defines the height of the element.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Resizing Elements by Changing Size Properties</i> on page 73</p>
HorizontalScrollbar	<p><b>Purpose:</b> Determines if a horizontal scroll bar appears on a list box control to allow the user to scroll the list box items horizontally.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> List Box</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Using a Horizontal Scroll Bar in a List Box Control</i> on page 143</p>
Image*	<p><b>Purpose:</b> Defines the image that is rendered in the element. Any image format supported by the application can be used.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Image</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Selecting a Different Image</i> on page 121</p>

Property	Purpose, category, usage and further information
ImageAlignment*	<p><b>Purpose:</b> Controls the location of the image relative to the bounding rectangle of the graphic. This property is only applicable when the ImageStyle is set to Normal.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Image</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Image Alignment</i> on page 120</p>
ImageStyle	<p><b>Purpose:</b> Defines how the image is rendered relative to its bounding rectangle.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Button, Image</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Image Display Mode</i> on page 119</p>
IntegralHeight	<p><b>Purpose:</b> Determines if the List Box size is an integral multiple of the Font Size so that a finite number of items fit in it without being clipped.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Combo Box, List Box</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Avoiding Clipping of Items in the Simple Combo Box Control</i> on page 138</p>
Layout*	<p><b>Purpose:</b> Defines the way the radio buttons are arranged in the group (Horizontal or Vertical).</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Radio Button Group</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Layout of the Radio Button Group Options</i> on page 134</p>

Property	Purpose, category, usage and further information
Locked	<p><b>Purpose:</b> Locks or unlocks the element's size, position, orientation and origin. Other properties that can have an affect on element size, position, orientation and origin are also locked. These are element-specific.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Locking and Unlocking Elements</i> on page 85</p>
MaxDropDownItems*	<p><b>Purpose:</b> Defines the maximum number of items the drop-down list shows.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Combo Box</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Maximum Number of Items to Appear in the Combo Box Drop-Down List</i> on page 138</p>
Multiline*	<p><b>Purpose:</b> Determines if the control shows several lines of text that automatically wrap up when reaching the right border of the control.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Edit Box</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Configuring the Text to Wrap in an Edit Box Control</i> on page 136</p>
Radius*	<p><b>Purpose:</b> Defines the radius of the corners of the Rounded Rectangle.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Rounded Rectangle</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Setting the Radius of Rounded Rectangles</i> on page 117</p>
ReadOnly*	<p><b>Purpose:</b> Determines if the user can type data into the edit box.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Edit Box</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Configuring the Text to be Read-Only in an Edit Box Control</i> on</p>

Property	Purpose, category, usage and further information
	page 136
RelativeAnchor*	<p><b>Purpose:</b> Relative anchor point of the source graphic. By default, this is 0,0.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Canvas</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Size Propagation and Anchor Points</i> on page 34</p>
RelativeOrigin	<p><b>Purpose:</b> Defines the relative origin as X, Y location. The location is relative to the center point of the element (0, 0).</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Group, Path, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Changing Points of Origin in the Properties Editor</i> on page 77</p>
ShowToday*	<p><b>Purpose:</b> Determines if today's date is shown on the calendar control.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Calendar</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Showing or Hiding Today's Date on a Calendar Control</i> on page 139</p>
Smoothing*	<p><b>Purpose:</b> When False the graphics are rendered normally, when True graphics are rendered with anti-aliasing which produces a smoother appearing graphic.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Canvas</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Radius of Rounded Rectangles</i> on page 117</p>

Property	Purpose, category, usage and further information
Start	<p><b>Purpose:</b> Defines the start of a line or H/V line as X, Y location.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Line, H/V Line</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting Start or End Points of a Line</i> on page 98</p>
StartAngle	<p><b>Purpose:</b> Defines the starting angle of an Arc, Pie or Chord. 0 is always the top of the graphic relative to its orientation. A positive number is clockwise from 0 and a negative number is counter clockwise from 0. If a negative number is used to set the property it is automatically converted to a positive value.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, 2 Point Arc, 3 Point Arc</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Changing Angles of Arcs, Pies and Chords</i> on page 124</p>
SweepAngle	<p><b>Purpose:</b> Defines the ending angle of the Arc, Pie or Chord. This angle is always measured from the start angle.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, 2 Point Arc, 3 Point Arc</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Changing Angles of Arcs, Pies and Chords</i> on page 124</p>
Tension	<p><b>Purpose:</b> Specifies how tightly the curve bends through the control points of the curve.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Closed Curve, Curve</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Changing the Tension of Curves and Closed Curves</i> on page 124</p>
Text	<p><b>Purpose:</b> Defines the unicode text that is shown by the element.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Button, Text, Text Box, Edit Box</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Setting the Displayed Text</i> on page 100</p>

Property	Purpose, category, usage and further information
TextFormat	<p><b>Purpose:</b> Defines the formatting string that is applied to the text when it is shown.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Button, Text, Text Box</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Setting the Text Display Format</i> on page 100</p>
Transparency	<p><b>Purpose:</b> Defines the transparency. A value of 0 means fully opaque and a value of 100 means fully transparent.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Group, Path, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Setting the Transparency Level of an Element</i> on page 111</p>
TransparentColor*	<p><b>Purpose:</b> Defines the RGB color value that is used as the transparent color.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Image</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Image Color Transparency</i> on page 120</p>
UpImage*	<p><b>Purpose:</b> Defines the image that is used in the button element when it is un-clicked or released.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Button</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Configuring Buttons with Images</i> on page 122</p>
Width	<p><b>Purpose:</b> Defines the width of the element.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Resizing Elements by Changing Size Properties</i> on page 73</p>

Property	Purpose, category, usage and further information
WordWrap	<p><b>Purpose:</b> When set to True, the text in the button or text box is formatted to fit as much text on a single line within the horizontal bounding area of the element and then continued to the next line. This continues as long as there is vertical space.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Button, Text Box</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Wrapping Text in Buttons</i> on page 122</p>
X	<p><b>Purpose:</b> Defines the left position of the element.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Moving Elements</i> on page 67</p>
Y	<p><b>Purpose:</b> Defines the top position of the element.</p> <p><b>Category:</b> Appearance</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Status, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Moving Elements</i> on page 67</p>

## Fill Style Group Properties

The following table contains a list of properties in the Fill Style property category used by the:

- Elements.
- Canvas.
- Element groups.
- Embedded graphics.

It describes their purpose, where they are used and where to find more information on how to use them.



An asterisk (\*) identifies properties that apply to only one type of element or the canvas, a group or an embedded graphic.

Property	Purpose, category, usage and further information
FillBehavior	<p><b>Purpose:</b> Determines how the Fill (Horizontal, Vertical or Both) should be applied to the element.</p> <p><b>Category:</b> Fill Style</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting Fill Behavior</i> on page 97</p>
FillColor	<p><b>Purpose:</b> Defines the fill style used for the filled portion of the element.</p> <p><b>Category:</b> Fill Style</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, List Box, Path</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting Fill Style</i> on page 96 and <i>Changing Background Color and Text Color of Windows Common Controls</i> on page 133</p>
FillOrientation	<p><b>Purpose:</b> Determines the orientation of the fill when the element orientation is any value other than 0.</p> <p><b>Category:</b> Fill Style</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting Fill Orientation</i> on page 97</p>
HorizontalDirection	<p><b>Purpose:</b> Determines the horizontal direction of the fill for the element. Can be "Right" or "Left".</p> <p><b>Category:</b> Fill Style</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting Horizontal Fill Direction and Percentage</i> on page 97</p>

Property	Purpose, category, usage and further information
HorizontalPercentFill	<p><b>Purpose:</b> Determines the percentage of horizontal fill for the element.</p> <p><b>Category:</b> Fill Style</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Setting Horizontal Fill Direction and Percentage on page 97</i></p>
TitleFillColor*	<p><b>Purpose:</b> Determines the background solid color in the title bar of the calendar control.</p> <p><b>Category:</b> Fill Style</p> <p><b>Used by:</b> Calendar</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting Title Fill Color and Text Color on a Calendar Control on page 140</i></p>
UnFilledColor	<p><b>Purpose:</b> Determines the element's unfilled area appearance.</p> <p><b>Category:</b> Fill Style</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting Unfilled Style on page 97</i></p>
VerticalDirection	<p><b>Purpose:</b> Defines the vertical direction of the fill. Can be "Top" or "Bottom".</p> <p><b>Category:</b> Fill Style</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting Vertical Fill Direction and Percentage on page 98</i></p>
VerticalPercentFill	<p><b>Purpose:</b> Determines the percentage of vertical fill for the element.</p> <p><b>Category:</b> Fill Style</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Button, Text Box, Path</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Setting Vertical Fill Direction and Percentage on page 98</i></p>

## Line Style Group Properties

The following table contains a list of properties in the Line Style property category used by the:

- Elements.
- Canvas.
- Element groups.
- Embedded graphics.

It shows their purpose, where they are used and where to find more information on how to use them.

Asterisk (\*) marks properties that are specific to only one type of element or the canvas, a group or an embedded graphic.

Property	Purpose, category, usage and further information
EndCap	<p><b>Purpose:</b> Defines the cap used at the end of the line of an open element.</p> <p><b>Category:</b> Line Style</p> <p><b>Used by:</b> Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting Line End Shape and Size</i> on page 118</p>
LineColor	<p><b>Purpose:</b> Defines the color and affects of the line or border.</p> <p><b>Category:</b> Line Style</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Text Box, Path</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Line Style</i> on page 99</p>
LinePattern	<p><b>Purpose:</b> Defines the pattern of the line or border.</p> <p><b>Category:</b> Line Style</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Text Box, Path</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Line Pattern</i> on page 99</p>
LineWeight	<p><b>Purpose:</b> Determines the weight of the element's line or border. A value of 0 means that there is no line or border.</p> <p><b>Category:</b> Line Style</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Text Box, Path</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Setting the Line Weight</i> on page 99</p>

Property	Purpose, category, usage and further information
StartCap	<p><b>Purpose:</b> Defines the cap used at the start of the line of an open graphic.</p> <p><b>Category:</b> Line Style</p> <p><b>Used by:</b> Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting Line End Shape and Size</i> on page 118</p>

## Text Style Group Properties

The following table contains a list of properties in the Text Style property category used by the:

- Elements.
- Canvas.
- Element groups.
- Embedded graphics.

It shows their purpose, where they are used and where to find more information on how to use them.

An asterisk (\*) indicates properties that are specific to only one type of element or the canvas, a group or an embedded graphic.

Property	Purpose, category, usage and further information
Alignment	<p><b>Purpose:</b> Controls the location of the text relative to the bounding rectangle of the element.</p> <p><b>Category:</b> Text Style</p> <p><b>Used by:</b> Button, Text, Text Box</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Text Alignment</i> on page 101</p>
Caption*	<p><b>Purpose:</b> Defines the text shown on the Check Box at design time and at run time when the caption property is not bound to a reference in the checkbox animation panel.</p> <p><b>Category:</b> Text Style</p> <p><b>Used by:</b> Check Box</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Caption Text of a Check Box Control</i> on page 135</p>

Property	Purpose, category, usage and further information
Font	<p><b>Purpose:</b> Defines the basic text font as defined by the operating system.</p> <p><b>Category:</b> Text Style</p> <p><b>Used by:</b> Button, Text, Text Box, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Text Font</i> on page 100</p>
TextColor	<p><b>Purpose:</b> Defines the color and affects applied to the text.</p> <p><b>Category:</b> Text Style</p> <p><b>Used by:</b> Button, Text, Text Box, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Text Color</i> on page 101 and <i>Changing Background Color and Text Color of Windows Common Controls</i> on page 133</p>
TitleTextColor*	<p><b>Purpose:</b> Determines the text solid color in the title bar of the calendar control.</p> <p><b>Category:</b> Text Style</p> <p><b>Used by:</b> Calendar</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting Title Fill Color and Text Color on a Calendar Control</i> on page 140</p>
TrailingTextColor*	<p><b>Purpose:</b> Determines the text solid color of the text for the trailing days. The trailing days are days outside the current month.</p> <p><b>Category:</b> Text Style</p> <p><b>Used by:</b> Calendar</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Text Color for Trailing Dates in a Calendar Control</i> on page 140</p>

## Runtime Behavior Group Properties

The following table contains a list of properties in the Runtime Behavior property category used by the:

- Elements.
- Canvas.
- Element groups.
- Embedded graphics.

The table shows the purpose of Runtime Behavior properties, where they are used, and where to find more information on how to use them.

Asterisk (\*) marks properties that are specific to only one type of element or the canvas, a group or an embedded graphic.

Property	Purpose, category, usage and further information
Enabled	<p><b>Purpose:</b> When set to True enables the element at run time and allows the user to interact with it. If the property is set to False the user cannot use the mouse or keyboard to interact with the element. Data changes as a result of an animation or script still execute.</p> <p><b>Category:</b> Runtime Behavior</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Enabling and Disabling Elements for Run-Time Interaction</i> on page 112</p>
Language	<p><b>Purpose:</b> Defines the current language of the graphic.</p> <p><b>Category:</b> Runtime Behavior</p> <p><b>Used by:</b> Embedded Symbol</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> Selecting the Language for a Symbol.</p>
LanguageID	<p><b>Purpose:</b> Defines the current language ID of the graphic.</p> <p><b>Category:</b> Runtime Behavior</p> <p><b>Used by:</b> Embedded Symbol</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> Selecting the Language for a Symbol.</p>
MultiplePopupsAllowed*	<p><b>Purpose:</b> If False, ShowSymbol animations only show within a single dialog window no matter how many animations are invoked and regardless of how the animations are configured. If True, ShowSymbol animations show in separate dialog windows.</p> <p><b>Category:</b> Runtime Behavior</p> <p><b>Used by:</b> Canvas</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Setting the Radius of Rounded Rectangles</i> on page 117</p>

Property	Purpose, category, usage and further information
OwningObject*	<p><b>Purpose:</b> Used as the object reference to replace all "Me." references in expressions and scripts. Everywhere there is a "Me." reference this object reference is used instead. The object name can be set either using a tag or hierarchical name of an AutomationObject.</p> <p><b>Category:</b> Runtime Behavior</p> <p><b>Used by:</b> Embedded Symbol</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Detecting and Editing the Containing Object Instance</i> on page 243</p>
Scripts*	<p><b>Purpose:</b> Defines a collection of scripts configured for the graphic.</p> <p><b>Category:</b> Runtime Behavior</p> <p><b>Used by:</b> Canvas</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Adding and Maintaining Graphic Scripts</i> on page 215</p>
SymbolReference*	<p><b>Purpose:</b> Contains the exact location that the Embedded Symbol is linked to. This can help the user in locating the original definition for editing purposes.</p> <p>This property is always disabled.</p> <p><b>Category:</b> Runtime Behavior</p> <p><b>Used by:</b> Embedded Symbol</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Detecting the Source Graphic of an Embedded Graphic</i> on page 240</p>
TabOrder	<p><b>Purpose:</b> Defines the tab order for the element. The tab order is only used when navigating by the keyboard. This property is valid only when the TabStop property is set to true.</p> <p><b>Category:</b> Runtime Behavior</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Editing the Tab Order of an Element</i> on page 113</p>

Property	Purpose, category, usage and further information
TabStop	<p><b>Purpose:</b> Determines if the element can be navigated to and can receive focus at run time.</p> <p><b>Category:</b> Runtime Behavior</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Editing the Tab Order of an Element</i> on page 113</p>
TreatAsIcon	<p><b>Purpose:</b> If this property is set to False, the animations defined on the graphics within the group or embedded graphic take precedence over an animation defined on the group or embedded graphic. If there are no animations or the user clicked on an area of the group or embedded graphic that does not have an animation, then the group or embedded graphic animation executes.</p> <p>If the property is set to True, only the animation on the group or embedded graphic is executed. The interactive animations within the group or embedded graphic never execute.</p> <p><b>Category:</b> Runtime Behavior</p> <p><b>Used by:</b> Group, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Editing the Embedded Graphic</i> on page 239</p>
Visible	<p><b>Purpose:</b> Determines the visibility of the element. This property is configured at design time and used only at runtime. At design time all elements are visible irrespective of this setting.</p> <p><b>Category:</b> Runtime Behavior</p> <p><b>Used by:</b> Rectangle, Rounded Rectangle, Ellipse, Polygon, Closed Curve, 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc, Button, Text, Text Box, Image, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box, Group, Path, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> Yes</p> <p><b>Info:</b> <i>Changing the Visibility of Elements</i> on page 113</p>

## Custom Properties Group Properties

The following table contains a list of properties in the Custom Properties property category used by the:

- Elements.
- Canvas.
- Element groups.



- Embedded graphics.

It shows their purpose, where they are used and where to find more information on how to use them.

The Custom Properties group contains also any other custom property you define.

---

<b>Property</b>	<b>Purpose, category, usage and further information</b>
CustomProperties	<p><b>Purpose:</b> The collection of CustomProperties defined by the graphic.</p> <p><b>Category:</b> Custom Properties</p> <p><b>Used by:</b> Canvas, Embedded Symbol</p> <p><b>Can be read by script at run time:</b> No</p> <p><b>Info:</b> <i>Using Custom Properties</i> on page 145</p>

---

## Order of Precedence for Property Styles

The order of precedence for property styles from high to low is:

1. Quality and Status
2. Element Style Animation
3. Style Animations
4. Group-level Element Style
5. Element-level Element Style
6. Local element-level style



# APPENDIX B

## Windows Common Control List Methods

### Overview of Windows Common Control List Methods

You can use the methods of the Windows common controls to manipulate the controls at run time by using them in scripting.

The following table contains a list of methods you can use in scripting to:

- Load and save the contents of the Edit Box control from and to a file.
- Manipulate items in the lists of the List Box control and Combo Box control.
- Manipulate items in the lists of the List Box control and Combo Box control.

Method	Purpose, syntax and information
AddItem	<p><b>Purpose:</b> Add an item (coerced to String) to the list. If the list is sorted, then the new item is inserted at the right position and selected. If the list is unsorted, the item is added to the bottom of the list.</p> <p><b>Used by:</b> Combo Box, List Box</p> <p><b>Note:</b> This function does not work when using an Enum or Array to populate the List Box.</p> <p><b>Syntax:</b> ControlName.AddItem(CaptionString);</p> <p><b>Info:</b> <i>Adding and Inserting Items into a List</i> on page 224</p>
Clear	<p><b>Purpose:</b> Removes all items from the List. If the list is bound, it clears the bound reference (array or enum).</p> <p><b>Note:</b> This function does not work when using an Enum or Array to populate the List Box.</p> <p><b>Used by:</b> Combo Box, List Box</p> <p><b>Syntax:</b> ControlName.Clear();</p> <p><b>Info:</b> <i>Deleting Items from a List</i> on page 225</p>
DeleteItem	<p><b>Purpose:</b> Accepts an index as a parameter and removes that item from the list. The first item in the list has an index of 0.</p> <p><b>Used by:</b> Combo Box, List Box</p> <p><b>Syntax:</b> ControlName.DeleteItem(Index);</p> <p><b>Info:</b> <i>Deleting Items from a List</i> on page 225</p>
DeleteSelection	<p><b>Purpose:</b> Delete the currently selected item from the list.</p> <p><b>Used by:</b> Combo Box, List Box</p> <p><b>Syntax:</b> ControlName.DeleteSelection();</p> <p><b>Info:</b> <i>Deleting Items from a List</i> on page 225</p>

Method	Purpose, syntax and information
FindItem	<p><b>Purpose:</b> Accepts a string as a parameter and returns the index of the first item that matches the string. The first item in the list has an index of 0.</p> <p><b>Used by:</b> Combo Box, List Box</p> <p><b>Syntax:</b> ControlName.FindItem(SearchString);</p> <p><b>Info:</b> <i>Finding an Item in a List</i> on page 225</p>
GetItem	<p><b>Purpose:</b> Returns the item associated with an index supplied as a parameter to this function. The first item in the list has an index of 0.</p> <p><b>Used by:</b> Combo Box, List Box</p> <p><b>Syntax:</b> ItemCaption = ControlName.GetItem(Index);</p> <p><b>Info:</b> <i>Reading the Caption of a Selected Item in a List</i> on page 225</p>
InsertItem	<p><b>Purpose:</b> Inserts the supplied string after the current selection in the List. Does not work if list is sorted.</p> <p><b>Used by:</b> Combo Box, List Box</p> <p><b>Syntax:</b> ControlName.InsertItem(String);</p> <p><b>Info:</b> <i>Adding and Inserting Items into a List</i> on page 224</p>
SetItemData	<p><b>Purpose:</b> Associates a value with an item in the list which index is provided to the function. The first item in the list has an index of 0.</p> <hr/> <p><b>Note:</b> This function only works when UseValuesAsItems is set to false. It does not work when using an Enum or Array to populate the List Box control.</p> <hr/> <p><b>Used by:</b> Combo Box, List Box</p> <p><b>Syntax:</b> ControlName.SetItemData(Index,Value);</p> <p><b>Info:</b> <i>Associating Items with Values in a List</i> on page 226</p>
GetItemData	<p><b>Purpose:</b> Returns the value associated with the item in the list which index is supplied to the function. The first item in the list has an index of 0.</p> <hr/> <p><b>Note:</b> This function only works when UseValuesAsItems is set to false. It does not work when using an Enum or Array to populate the List Box control.</p> <hr/> <p><b>Used by:</b> Combo Box, List Box</p> <p><b>Syntax:</b> Value = ControlName.GetItemData(Index);</p> <p><b>Info:</b> <i>Associating Items with Values in a List</i> on page 226</p>

---

<b>Method</b>	<b>Purpose, syntax and information</b>
LoadList	<p><b>Purpose:</b> Loads a list of strings from a file which name is passed as parameter to the function. The default location for files is the users folder, for example: c:\documents and settings\username.</p> <hr/> <p><b>Note:</b> The LoadList method does not work when using an Enum or Array to populate the List Box control.</p> <hr/> <p><b>Used by:</b> Combo Box, List Box</p> <p><b>Syntax:</b> ControlName.LoadList(FileName);</p> <p><b>Info:</b> <i>Loading and Saving Item Lists</i> on page 226</p>
LoadText	<p><b>Purpose:</b> Loads a text from a file into the Edit Box control. The default location for files is the users folder, for example: c:\documents and settings\username.</p> <p><b>Used by:</b> Edit Box</p> <p><b>Syntax:</b> ControlName.LoadText(FileName);</p> <p><b>Info:</b> <i>Configuring Edit Box Methods</i> on page 223</p>
SaveList	<p><b>Purpose:</b> Save a list to a file which name is passed as parameter to the function. The default location for files is the users folder, for example: c:\documents and settings\username.</p> <p><b>Used by:</b> Combo Box, List Box</p> <p><b>Syntax:</b> ControlName.SaveList(FileName);</p> <p><b>Info:</b> <i>Loading and Saving Item Lists</i> on page 226</p>
SaveText	<p><b>Purpose:</b> Saves the current text in the Edit Box control to a file. The default location for files is the users folder, for example: c:\documents and settings\username.</p> <p><b>Used by:</b> Edit Box</p> <p><b>Syntax:</b> ControlName.SaveText(FileName);</p> <p><b>Info:</b> <i>Configuring Edit Box Methods</i> on page 223</p>

---



# APPENDIX C

## QuickScript References

### Script Functions

This section describes the script functions available in the HMI/SCADA development environment. The function documentation is organized into a set of folders that represents the same organization of the functions in the Script Function Browser.

Also provided are additional references for standard QuickScript .NET variables, control structures, and operators.

Other Microsoft .NET script functions, are not documented. Refer to Microsoft .NET documentation for descriptions of the functions.

### Graphic Client Functions

Use graphic client functions to hide and show symbols, open and close popup windows, log in and log off users, or to query custom properties contained in a symbol.

#### GetCPQuality()

Returns the Quality value of a custom property. This function is available within any Industrial Graphics client script, but may not be supported by your HMI. For more information, consult your HMI documentation.

##### Syntax

```
Int GetCPQuality(String name)
```

Where *String name* is the name of the custom property whose quality is to be retrieved.

This script function takes the name of a custom property on the symbol. This argument is of type string and it can be a reference or a constant.

If the custom property is type constant, GOOD is the quality always returned.

---

**Note:** For use with custom properties only. It does not apply to HMI tags.

---

##### Return Value

The GetCPQuality() script function returns a value 0-255 of type Integer, as per the OPC quality standard. 192 is GOOD.

##### Example

```
cp2 = GetCPQuality("cp1");
```

Where cp1 and cp2 are custom properties and the data type of cp2 is Integer.

#### GetCPTimeStamp()

Returns the time stamp of a custom property. This function is available within any Industrial Graphics client script.

##### Syntax

```
DateTime GetCPTimeStamp(String name)
```

Where *String name* is the name of the custom property whose time stamp is to be retrieved.

This script function takes the name of a custom property on the symbol. This argument is of type string and it can be a reference or a constant.

---

**Note:** For use with custom properties only. It does not apply to HMI tags.

---

### Return Value

The GetCPTimeStamp() script function returns the time stamp of the custom property's current value of type DateTime. If the custom property value is a constant, then the return value is the time the value was created.

Example

```
cp2 = GetCPTimeStamp("cp1");
```

Where cp1 and cp2 are custom properties and the data type of cp2 is DateTime.

## HideGraphic()

Closes an open graphic pop-up window shown in the ShowGraphic() script with the given identity name.

The HideGraphic() function has been extended to close HMI Windows identified with a given identity name. This function is available within any Industrial Graphics client script.

### Category

Graphic Client

### Syntax

```
HideGraphic(string identity);
```

### Parameter

*Identity*

The unique name of the instance that shows the graphic.

### Examples

```
HideGraphic("i1");
```

Where "i1" is string Identity.

```
HideGraphic("<HMName>:Window1");
```

Where "<HMName>1" is the string identity.

### See Also

*ShowGraphic()* on page 337, *HideSelf()* on page 336

## HideSelf()

Closes the displayed graphic or layout for which this script is configured. This script function is available within any Industrial Graphics client script.

### Category

Graphic Client

### Syntax

```
HideSelf();
```

### Remarks

For an Industrial Graphics script, you must call the script function within the symbol to hide the popup.



**Example**

```
HideSelf ();
```

**See Also**

*ShowGraphic()* on page 337, *HideGraphic()* on page 336

**Logoff()**

Action script that automatically logs off the current user from a ViewApp.

Action scripts are graphic animations that are triggered by a user action such as a mouse click.

**Category**

Miscellaneous

**Syntax**

```
LogOff();
```

**Parameter**

None

**Trigger**

On Left-Click/Key/Touch Down

**Additional Information**

A log off button can be added that uses the Logoff() method to allow the user to log off from the ViewApp.

**Example**

```
Logoff ();
```

**See Also**

*ShowLoginDialog()* on page 347

**ShowGraphic()**

Shows a graphic within a pop-up window. The ShowGraphic() function has been extended to call InTouch HMI Windows and AVEVA OMI layouts. This function is available within any Industrial Graphics client script or OMI layout script.

**Category**

Graphic Client

**Syntax****Show a graphic within a pop-up window**

```
Dim graphicInfo as aaGraphic.GraphicInfo;
graphicInfo.Identity = "<Identity>";
graphicInfo.GraphicName = "<SymbolName>";
ShowGraphic ( graphicInfo );
```

**Call an HMI window**

```
Dim graphicInfo as aaGraphic.GraphicInfo;
graphicInfo0.Identity = "<<HMName>:WindowName>";
ShowGraphic ( graphicInfo );
```

**Parameter***GraphicInfo***Data Type**

aaGraphic.GraphicInfo

**Examples****Show graphic within a pop-up window**

```
ShowGraphic (graphicInfo);
```

**Show an HMI window**

```
Dim graphicInfo0 as aaGraphic.GraphicInfo;
graphicInfo0.Identity = "<HMName>:Window1";
ShowGraphic( graphicInfo0 );
```

**aaGraphic.GraphicInfo Properties**

Any string properties can be a concatenation of strings and/or custom properties.

*Identity*

A unique name that identifies which instance has opened the graphic.

**Data Type**

String

**Additional Information**

Mandatory

The same Identity is used in the HideGraphic() script function to close the pop-up window.

**Valid Range**

The name cannot contain more than 329 characters.

The name must contain at least one letter.

Valid characters are alphanumeric and special characters (\$, #, \_).

**Example**

```
graphicInfo.Identity = "i1";
```

*GraphicName*

The name of the graphic to show.

**Data Type**

String

**Valid Range**

The name cannot contain more than 329 characters.

The name must contain at least one letter.

Valid characters are alphanumeric and special characters (\$, #, \_).

**Additional Information**

Mandatory

Browse using the **Display Galaxy Browser** or directly type the graphic name.

Galaxy name can come from:

- Graphic Toolbox, for example:  
"Symbol\_001"
- Instances, absolute or hierarchical, for example:  
"Userdefined\_001.Symbol1", "Userdefined\_001.Pump\_001.S1"
- Relative reference, for example:  
"Me.Symbol\_001"

Use an absolute name to specify the symbol name and owning object if you are using an Object Wizard with Symbol Wizard custom property selections. This allows the correct symbol configuration to be shown for the instance. See *Owning Object*, below, for more information.

If you type any invalid character or exceed the character limit, the system shows a warning message at run time. There is no validation at design time.

The graphic name can be a concatenation of constant strings and reference strings. For

example: "Pump\_001" + ".Symbol\_001"; cp1 + ".Symbol\_001", where the value of cp1 = "Pump\_001"; or Obj1.Str1 + ".Symbol\_001", where the value of Obj.Str1 = "Pump\_001".

## Examples

### Graphic Toolbox Reference

```
graphicInfo.GraphicName = "S1";
```

### Absolute Reference

```
graphicInfo.GraphicName = "OwningObjectName.SymbolName";
```

### *Owning Object*

The owning object of the graphic shown by the ShowGraphic() script function.

### Data Type

String

### Default Value

Empty

### Additional Information

Optional

Can be a concatenation of constant strings and reference strings.

Can be browsed using the **Display Automation Object Browser**, or you can type the name of the owning object.

---

**Note:** The *OwningObject* property sets references for the graphic, but is not associated with the *GraphicName* property if the symbol is part of an Object Wizard. Therefore, if you are scripting a symbol with an owning object, specify the owning object name as part of the *GraphicName* property, for example, UserDefined\_001.Pump\_001.

---

### Example

```
graphicInfo.OwningObject = "UserDefined_001";
```

### *HasTitleBar*

Determines if the graphic is shown with a title bar.

### Data Type

Boolean

### Default Value

True

### Example

```
graphicInfo.HasTitleBar = false;
```

### *WindowTitle*

Specifies the title shown in the window title bar.

### Data Type

String

### Default Value

Empty

**Valid Range**

Limit 1024 characters

**Additional Information**

Can be a constant string, a reference, or an expression.

If you change the owning object for an AutomationObject graphic, the window title is updated accordingly.

If the WindowTitle parameter is empty, the value of the Identity parameter is shown on the title bar.

**Example**

```
graphicInfo.WindowTitle = "Graphic01";
```

*WindowType*

Specifies whether window type is modal or modeless.

**Data Type**

Enum

**Default Value**

Modeless

**Valid Range**

0, 1

**Enumerations**

WindowType	Integer
Modal	0
Modeless	1

**Examples**

```
graphicInfo.WindowType = aaGraphic.WindowType.<windowtype>;
graphicInfo.WindowType = 1;
```

*HasCloseButton*

Determines if the pop-up window has a close button.

**Data Type**

Boolean

**Default Value**

True

**Example**

```
graphicInfo.HasCloseButton = false;
```

*Resizable*

Determines if the pop-up window is resizable.

**Data Type**

Boolean

**Default Value**

False

**Example**

```
graphicInfo.Resizable = true;
```

*WindowLocation*

Specifies the location of the pop-up window.

**Data Type**

Enum

**Default Value**

Center

**Valid Range**

One of 0–12

**Enumerations**

<b>WindowLocation</b>	<b>Integer</b>
Center	0
Above	1
TopLeftCorner	2
Top	3
TopRightCorner	4
LeftOf	5
LeftSide	6
RightSide	7
RightOf	8
BottomLeftCorner	9
Bottom	10
BottomRightCorner	11
Below	12

**Additional Information**

If you have selected Desktop as the window relative position, Above, LeftOf, RightOf, and Below are invalid.

For more information about the behavior of the WindowLocation parameter, see "Working with the Show/Hide Graphics Script Functions," in the *Creating and Managing Industrial Graphics User Guide*.

**Examples**

```
graphicInfo.WindowLocation = aaGraphic.WindowLocation.<WindowLocation>;
graphicInfo.WindowLocation = 1;
```

*WindowRelativePosition*

Specifies the relative position of the pop-up window.

**Data Type**

Enum

**Default Value**

Desktop

**Valid Range**

One of 0–8

**Enumerations**

WindowRelativePosition	Integer
Desktop	0
Window	1
ClientArea	2
ParentGraphic	3
ParentElement	4
Mouse	5
DesktopXY	6
WindowXY	7
ClientAreaXY	8

**Examples**

```
graphicInfo.WindowRelativePosition =
aaGraphic.WindowRelativePosition.<WindowRelativePosition>;
graphicInfo.WindowRelativePosition = 1;
```

*RelativeTo*

Specifies the size of the pop-up window relative to the graphic, desktop, or customized width and height.

**Data Type**

Enum

**Default Value**

Graphic

**Valid Range**

One of 0–2

**Enumerations**

RelativeTo	Integer
Graphic	0
Desk Top	1
CustomizedWidthHeight	2

**Additional Information**

If you enter `aaGraphic.RelativeTo.CustomizedWidthHeight`, you can include the values of the height and width in the script. Otherwise, the default values are used.

**Examples**

```
graphicInfo.RelativeTo = aaGraphic.RelativeTo.<RelativeTo>;  
graphicInfo.RelativeTo = 1;
```

X

The horizontal position of the pop-up window.

**Data Type**

Integer

**Default Value**

0

**Valid Range**

-2,147,483,648 through 2,147,483,647

**Additional Information**

If X is beyond the integer range, an overflow message appears in the Logger at run time. This parameter is applicable only if the value of the `WindowRelativePosition` parameter is `DesktopXY`, `WindowXY`, or `ClientAreaXY`. Unlike the `ShowSymbol` animation, there is no boundary for this value.

**Examples**

```
graphicInfo.X = 100;
```

Y

Specifies the vertical position of the pop-up window.

**Data Type**

Integer

**Default Value**

0

**Valid Range**

-2,147,483,648 through 2,147,483,647

**Additional Information**

If Y is beyond integer range, a proper overflow message will appear in the Logger at run time. This value is applicable only if `WindowRelativePosition` is `DesktopXY`, `WindowXY`, or `ClientAreaXY`.

**Unlike the `ShowSymbol` animation, there is no boundary for this value.**

**Examples**

```
graphicInfo.Y = 100;
```

*Width*

Specifies the width of the pop-up window.

**Data Type**

Integer

**Default Value**

100

**Valid Range**

0–10000

**Additional Information**

Applicable only if `RelativeTo` is `CustomizedWidthHeight`

You can specify either the height or the width of the pop-up window. The system calculates the other, based on the aspect ratio of the symbol.

If you enter an out-of-boundary value, the system shows an "Out of range" message at run time. If the value > 10000, it is set at 10000. If the value < 0, it is set at 0.

**Examples**

```
graphicInfo.width = 500;
```

*Height*

Specifies the height of the pop-up window.

**Data Type**

Integer

**Default Value**

100

**Valid Range**

0–10000

**Additional Information**

Applicable only if RelativeTo is the value of the CustomizedWidthHeight parameter.

You can specify either the height or the width of the pop-up window. The system calculates the other, based on the aspect ratio of the symbol.

If you enter an out-of-boundary value, the system shows an "Out of range" message at run time. If the value > 10000, it is set at 10000. If the value < 0, it is set at 0.

**Examples**

```
graphicInfo.height = 500;
```

*TopMost*

Sets a value that indicates whether the ShowGraphic appears in the top most z-order window. A ShowGraphic whose Topmost property is set to true appears above all windows whose TopMost properties are set to false (same as Windows Task Manager).

**Data Type**

Boolean

**Default Value**

False

**Additional Information**

ShowGraphic windows whose Topmost properties are set to true appear above all windows whose Topmost properties are set to false. In a group of windows that have the Topmost property set to true, the active window is the topmost window.

---

**Note:** Do not create scripts that launch a non-TopMost Modal dialog from a TopMost dialog. Users will not be able to interact with the View if the Modal dialog is completely hidden by any TopMost window.

---

**Example**

```
graphicInfo.TopMost = true;
```

*ScalePercentage*

Sets the scaling percentage of the pop-up window and the graphic it contains.

**Data Type**

Integer

**Default Value**

100



**Valid Range**

0–1000

**Additional Information**

If you enter an out-of-boundary value, the system shows an "Out of range" message at run time. If the value > 1000, it is set at 1000. If the value < 0, it is set at 0.

**Examples**

```
graphicInfo.ScalePercentage = 150;
```

*KeepOnMonitor*

Specifies that a pop-up window should appear entirely within the boundaries of an application window.

**Data Type**

Boolean

**Default Value**

True

**Example**

```
graphicInfo.KeepOnMonitor = true;
```

*StretchGraphicToFitWindowSize*

Determines if the graphic is scaled to the current size of the pop-up window.

**Data Type**

Boolean

**Default Value**

True

**Additional Information**

Applicable only if the value of the ScalePercentage parameter is greater than 100.

**Examples**

```
graphicInfo.StretchGraphicToFitWindowSize = false;
```

*StretchWindowToScreenWidth*

Determines if the pop-up window is scaled to the same width as the screen.

**Data Type**

Boolean

**Default Value**

False

**Additional Information**

Applicable only if the WindowRelativePosition parameter is Desktop, Window, Client Area, ParentGraphic, or ParentElement.

**Examples**

```
graphicInfo.StretchWindowToScreenWidth = true;
```

*StretchWindowToScreenHeight*

Determines if the pop-up window is scaled to the same height as the screen.

**Data Type**

Boolean

**Default Value**

False

**Additional Information**

Applicable only if the WindowRelativePosition parameter is Desktop, Window, Client Area, ParentGraphic, or ParentElement.

**Examples**

```
graphicInfo.StretchWindowToScreenHeight = true;
```

**CustomProperties**

Sets the custom properties of the symbol being shown.

**Data Type**

CustomPropertyValuePair[] array

**Additional Information**

The first three parameters are custom property name, value, and IsConstant.

Both custom property and the value can be a constant string, reference, or concatenation of strings. If the parameter IsConstant = True, the value is treated as a constant. Otherwise, the value is treated as a reference.

The array index starts at 1.

**Examples**

```
Dim cpValues [4] as aaGraphic.CustomPropertyValuePair;
cpValues[1] = new aaGraphic.CustomPropertyValuePair("CP1", 20, true);
cpValues[2] = new aaGraphic.CustomPropertyValuePair("CP2", Pump.PV.TagName,
true);
cpValues[3] = new aaGraphic.CustomPropertyValuePair("CP3", "CP"+var1, CP2 +
"001" + ".Speed", true);
cpValues[4] = new aaGraphic.CustomPropertyValuePair("CP3",
"<HMName>:Tag1", false);
graphicInfo.CustomProperties = cpValues;
```

**Remarks**

Any parameter that has default value in the GraphicInfo is optional. If no input value specified for these parameters, the default values are used at run time. Any parameter except the Enum data type can be a constant, reference, or expression.

For more information, see "Working with the Show/Hide Graphics Script Functions" in the *Industrial Graphic Editor User Guide*.

**Examples for ShowGraphic****Basic script example:**

```
Dim graphicInfo as aaGraphic.GraphicInfo;
graphicInfo.Identity = "Script_001";
graphicInfo.GraphicName = "Symbol_001";
ShowGraphic( graphicInfo );
```

**Advanced script example:**

```
Dim graphicInfo as aaGraphic.GraphicInfo;
Dim cpValues [2] as aaGraphic.CustomPropertyValuePair;
cpValues[1] = new aaGraphic.CustomPropertyValuePair("CP1", 20, true);
cpValues[2] = new aaGraphic.CustomPropertyValuePair("CP2", "Pump.PV.TagName",
false);
graphicInfo.Identity = "i1";
graphicInfo.GraphicName = "S1";
graphicInfo.OwningObject = "UserDefined_001";
graphicInfo.WindowTitle = "Graphic01";
graphicInfo.Resizable = false;
graphicInfo.CustomProperties=cpValues;
ShowGraphic( graphicInfo );
```

Where "i1" is string Identity and the symbol "S1" contains custom property CP1 and CP2.

### See Also

*HideSelf()* on page 336

## ShowLoginDialog()

Action script that shows a login dialog box with fields to enter a username and password. A typical login interface includes a login button that is selected by the user to show the **Login** dialog box with fields to enter a username and password.

Action scripts are graphic animations that are triggered by a user action such as a mouse click.

### Category

Miscellaneous

### Syntax

```
ShowLoginDialog() ;
```

### Parameter

None

### Trigger

On Left-Click/Key/Touch Down

### Additional Information

A log off button can be added that uses the *Logoff()* method to allow the user to log off from the ViewApp.

### Example

```
ShowLoginDialog () ;
```

### See Also

*Logoff()* on page 337

## Math Functions

Use math functions to return the answer to the specified mathematical expression.

In QuickScript, all mathematical operations are calculated internally as double, regardless of the operand data type. Following standard mathematical rules, the result is always rounded in division operations to maintain accuracy. Rounding only occurs on the end result, not intermediate values, and the quotient will match the target data type. This is the standard methodology for SCADA and DCS systems, and provides the data integrity, precision retention, time stamps, and overall data quality propagation and aggregation needed for these systems.

If you want to round at each step instead of only at the final result, you can leverage the support built into QuickScript for .NET libraries and utilize the *System.Math.Floor* and *System.Math.Round* methods to explicitly round the intermediate steps. As an example, consider the following script:

```
dim dividend as integer;
dim divisor as integer;
dim quotient as integer;
dim remainder as integer;
dividend = 8;
divisor = 3;
LogMessage("Value of dividend = " + dividend);
LogMessage("Value of divisor = " + divisor);
quotient = dividend/divisor;
LogMessage("Value of quotient = " + quotient);
remainder = dividend mod divisor;
LogMessage ("Value of remainder = " + remainder);
dividend = divisor*quotient +remainder;
LogMessage ("Value of dividend = " + dividend);
```

The result is:  $8 / 3 = 3$

If, instead, you want to drop the remainder (not rounding the final result to the nearest integer), you could add a call to the `Math.Floor` method and use the following:

```
dim dividend as integer;
dim divisor as integer;
dim quotient as integer;
dim remainder as integer;
dividend = 8;
divisor = 3;
LogMessage("Value of dividend = " + dividend);
LogMessage("Value of divisor = " + divisor);

// *** Add call to Math.Floor. This drops the remainder rather than rounding the
internal Double result to integer
quotient = System.Math.Floor(dividend/divisor);

LogMessage("Value of quotient = " + quotient);
remainder = dividend mod divisor;
LogMessage ("Value of remainder = " + remainder);
dividend = divisor*quotient +remainder;
LogMessage ("Value of dividend = " + dividend);
```

The result is:  $8 / 3 = 2$  (remainder 2)

## Abs()

Returns the absolute value (unsigned equivalent) of a specified number.

### Category

Math

### Syntax

```
Result = Abs ( Number );
```

### Parameter

*Number*

Any number or numeric attribute.

### Examples

```
Abs(14); ' returns 14
Abs(-7.5); ' returns 7.5
```

## ArcCos()

Returns an angle between 0 and 180 degrees whose cosine is equal to the number specified.

### Category

Math

### Syntax

```
Result = ArcCos( Number );
```

### Parameter

*Number*

Any number or numeric attribute with a value between -1 and 1 (inclusive).

### Examples

```
ArcCos(1); ' returns 0  
ArcCos(-1); ' returns 180
```

### See Also

*Cos()* on page 350, *Sin()* on page 353, *Tan()* on page 354, *ArcSin()* on page 349, *ArcTan()* on page 349

## ArcSin()

Returns an angle between -90 and 90 degrees whose sine is equal to the number specified.

### Category

Math

### Syntax

```
Result = ArcSin( Number );
```

### Parameter

*Number*

Any number or numeric attribute with a value between -1 and 1 (inclusive).

### Examples

```
ArcSin(1); ' returns 90  
ArcSin(-1); ' returns -90
```

### See Also

*Cos()* on page 350, *Sin()* on page 353, *Tan()* on page 354, *ArcCos()* on page 349, *ArcTan()* on page 349

## ArcTan()

Returns an angle between -90 and 90 degrees whose tangent is equal to the number specified.

### Category

Math

### Syntax

```
Result = ArcTan( Number );
```

### Parameter

*Number*

Any number or numeric attribute.

### Examples

```
ArcTan(1); ' returns 45
```

```
ArcTan(0); ' returns 0
```

**See Also**

*Cos()* on page 350, *Sin()* on page 353, *Tan()* on page 354, *ArcCos()* on page 349, *ArcSin()* on page 349

## Cos()

Returns the cosine of an angle in degrees.

**Category**

Math

**Syntax**

```
Result = Cos ( Number );
```

**Parameter**

*Number*

Any number or numeric attribute.

**Examples**

```
Cos(90); ' returns 0
```

```
Cos(0); ' returns 1
```

This example shows how to use the function in a math equation:

```
Wave = 50 * Cos(6 * Now().Second);
```

**See Also**

*Sin()* on page 353, *Tan()* on page 354, *ArcCos()* on page 349, *ArcSin()* on page 349, *ArcTan()* on page 349

## Exp()

Returns the result of the exponent *e* raised to a power.

**Category**

Math

**Syntax**

```
Result = Exp ( Number );
```

**Parameter**

*Number*

Any number or numeric attribute.

**Example**

```
Exp(1); ' returns 2.718...
```

## Int()

Returns the next integer less than or equal to a specified number.

**Category**

Math

**Syntax**

```
IntegerResult = Int ( Number );
```

**Parameter**

*Number*  
Any number or numeric attribute.

**Remarks**

When handling negative real (float) numbers, this function returns the integer farthest from zero.

**Examples**

```
Int(4.7); ' returns 4  
Int(-4.7); ' returns -5
```

**Log()**

Returns the natural log (base e) of a number.

**Category**

Math

**Syntax**

```
RealResult = Log( Number );
```

**Parameter**

*Number*  
Any number or numeric attribute.

**Remarks**

Natural log of 0 is undefined.

**Examples**

```
Log(100); ' returns 4.605...  
Log(1); ' returns 0
```

**See Also**

*LogN()* on page 352, *Log10()* on page 351

**Log10()**

Returns the base 10 log of a number.

**Category**

Math

**Syntax**

```
Result = Log10( Number );
```

**Parameter**

*Number*  
Any number or numeric attribute.

**Example**

```
Log10(100); ' returns 2
```

**See Also**

*Log()* on page 351, *LogN()* on page 352

## LogN()

Returns the values of the logarithm of x to base n.

### Category

Math

### Syntax

```
Result = LogN( Number, Base );
```

### Parameters

*Number*

Any number or numeric attribute.

*Base*

Integer to set log base. You could also specify an integer attribute.

### Remarks

Base 1 is undefined.

### Examples

```
LogN(8, 3); ' returns 1.89279
```

```
LogN(3, 7); ' returns 0.564
```

### See Also

[Log\(\)](#) on page 351, [Log10\(\)](#) on page 351

## Pi()

Returns the value of Pi.

### Category

Math

### Syntax

```
RealResult = Pi();
```

### Example

```
Pi(); ' returns 3.1415926
```

## Round()

Rounds a real number to a specified precision and returns the result.

### Category

Math

### Syntax

```
RealResult = Round( Number, Precision );
```

### Parameters

*Number*

Any number or numeric attribute.

*Precision*

Sets the precision to which the number is rounded. This value can be any number or a numeric attribute.



**Examples**

```
Round(4.3, 1); ' returns 4
Round(4.3, .01); ' returns 4.30
Round(4.5, 1); ' returns 5
Round(-4.5, 1); ' returns -4
Round(106, 5); ' returns 105
Round(43.7, .5); ' returns 43.5
```

**See Also**

*Trunc()* on page 354

**Sgn()**

Determines the sign of a value (whether it is positive, zero, or negative) and returns the result.

**Category**

Math

**Syntax**

```
IntegerResult = Sgn( Number );
```

**Parameter**

*Number*

Any number or numeric attribute.

**Return Value**

If the input number is positive, the result is 1. Negative numbers return a -1, and 0 returns a 0.

**Examples**

```
Sgn(425); ' returns 1;
Sgn(0); ' returns 0;
Sgn(-37.3); ' returns -1;
```

**Sin()**

Returns the sine of an angle in degrees.

**Category**

Math

**Syntax**

```
Result = Sin( Number );
```

**Parameter**

*Number*

Angle in degrees. Any number or numeric attribute.

**Examples**

```
Sin(90); ' returns 1;
Sin(0); ' returns 0;
```

This example shows how to use the function in a math expression:

```
wave = 100 * Sin ( 6 * Now().Second );
```

**See Also**

*Cos()* on page 350, *Tan()* on page 354, *ArcCos()* on page 349, *ArcSin()* on page 349, *ArcTan()* on page 349

## Sqrt()

Returns the square root of a number.

### Category

Math

### Syntax

```
RealResult = Sqrt ( Number );
```

### Parameter

*Number*

Any number or numeric attribute.

### Example

This example takes the value of me.PV and returns the square root as the value of x:

```
x=Sqrt (me.PV);
```

## Tan()

Returns the tangent of an angle given in degrees.

### Category

Math

### Syntax

```
Result = Tan ( Number );
```

### Parameter

*Number*

The angle in degrees. Any number or numeric attribute.

### Examples

```
Tan(45); ' returns 1;
```

```
Tan(0); ' returns 0;
```

This example shows how to use the function in a math expression:

```
Wave = 10 + 50 * Tan(6 * Now().Second);
```

### See Also

[Cos\(\)](#) on page 350, [Sin\(\)](#) on page 353, [ArcCos\(\)](#) on page 349, [ArcSin\(\)](#) on page 349, [ArcTan\(\)](#) on page 349

## Trunc()

Truncates a real (floating point) number by simply eliminating the portion to the right of the decimal point, including the decimal point, and returns the result.

### Category

Math

### Syntax

```
NumericResult = Trunc ( Number );
```

### Parameter

*Number*

Any number or numeric attribute.

**Remarks**

This function accomplishes the same result as placing the contents of a float type attribute into an integer type attribute.

**Examples**

```
Trunc(4.3); ' returns 4;
Trunc(-4.3); ' returns -4;
```

**See Also**

*Round()* on page 352

## Miscellaneous Functions

Functions in the miscellaneous group perform a variety of purposes, such as logging data or querying attributes.

### ActivateApp()

Restores, minimizes, maximizes, or closes another currently running Windows application.

**Category**

Miscellaneous

**Syntax**

```
ActivateApp( TaskName );
```

**Parameter**

*TaskName*

The task this function activates.

**Remarks**

*TaskName* is the exact text string, including spaces, that appears on the Task Bar or in Windows Task Manager. You can see the task name by opening Task Manager.

**Example**

```
ActivateApp("Calculator");
```

### Filtering Events

To get only specific events, filters can be introduced before getting events from the event service. The filtering should be done before the `StartRequestingEvent()` method is called.

The following datatypes are supported when filtering the events.

- Integer
- Float
- String
- Bool
- DateTime
- Double
- Short
- Array

The following table shows the comparison types that are supported for filtering events.

<b>Comparison Keyword</b>	<b>Description</b>
eq	Means EqualTo. Returns all the events matching the filtered criteria.
beginswith	Means StartsWith. Returns all the events matching the filtered criteria. Applies only to string data type filtering
lt	Means Lesser Than. Applies to all supported data types excluding string. It does not support arrays.
le	Means Lesser or Equal. Applies to all supported data types excluding string. It does not support arrays.
gt	Means Greater Than. Applies to all supported data types excluding string. It does not support arrays.
ge	Means Greater or Equal. Applies to all supported data types excluding string. It does not support arrays.
between	Checks will be made only to paired supplied values. Returns all the events matching the filtered criteria. It supports numeric and date data types.
neg, nbegins, nlt, nle, ngt, nge, nbetween	A keyword 'n' before the comparison keyword Means NOT of.

## DateTimeGMT()

Returns a number representing the number of days and fractions of days since January 1, 1970, in Coordinated Universal Time (UTC), regardless of the local time zone.

### Category

Miscellaneous

### Syntax

```
Result=DateTimeGMT ();
```

### Parameters

None

### Example

```
MessageTag = StringFromTime(DateTimeGMT() * 86400.0, 3);
```

## IsBad()

Returns a Boolean value indicating if the quality of the specified attribute is Bad.

### Category

Miscellaneous

### Syntax

```
BooleanResult = IsBad( Attribute1, Attribute2, ... );
```

### Parameter(s)

*Attribute1, Attribute2, ...AttributeN*

Names of one or more attributes for which you want to determine Bad quality. You can include a variable-length list of attributes.

## Return Value

If any of the specified attributes has Bad quality, then true is returned. Otherwise, false is returned.

## Examples

```
IsBad(TIC101.PV);  
IsBad(TIC101.PV, PIC102.PV);
```

## See Also

*IsGood()* on page 357, *IsInitializing()* on page 357, *IsUncertain()* on page 358, *IsUsable()* on page 358

## IsGood()

Returns a Boolean value indicating if the quality of the specified attribute is Good.

## Category

Miscellaneous

## Syntax

```
BooleanResult = IsGood( Attribute1, Attribute2, ... );
```

## Parameter(s)

*Attribute1, Attribute2, and so on*

Name of the attribute(s) for which you want to determine Good quality. You can include a variable-length list of attributes.

## Return Value

If all of the specified attributes have Good quality, then true is returned. Otherwise, false is returned.

## Examples

```
IsGood(TIC101.PV);  
IsGood(TIC101.PV, PIC102.PV);
```

## See Also

*IsBad()* on page 356, *IsInitializing()* on page 357, *IsUncertain()* on page 358, *IsUsable()* on page 358

## IsInitializing()

Returns a Boolean value indicating if the quality of the specified attribute is Initializing.

## Category

Miscellaneous

## Syntax

```
BooleanResult = IsInitializing( Attribute1, Attribute2, ... );
```

## Parameter(s)

*Attribute1, Attribute2, and so on*

Name of the attribute(s) for which to determine Initializing quality. You can include a variable-length list of attributes.

## Return Value

If any of the specified attributes has Initializing quality, then true is returned. Otherwise, false is returned.

## Examples

```
IsInitializing(TIC101.PV);  
IsInitializing(TIC101.PV, PIC102.PV);
```

**See Also**

*IsBad()* on page 356, *IsGood()* on page 357, *IsUncertain()* on page 358, *IsUsable()* on page 358

**IsUncertain()**

Returns a Boolean value indicating if the quality of the specified attribute is Uncertain.

**Category**

Miscellaneous

**Syntax**

```
BooleanResult = IsUncertain( Attribute1, Attribute2, ... );
```

**Parameter(s)**

*Attribute1, Attribute2, and so on*

Name of the attribute(s) to determine Uncertain quality. You can include a variable-length list of attributes.

**Return Value**

If all of the specified attributes have Uncertain quality, then true is returned. Otherwise, false is returned.

**Examples**

```
IsUncertain(TIC101.PV);  
IsUncertain(TIC101.PV, PIC102.PV);
```

**See Also**

*IsBad()* on page 356, *IsGood()* on page 357, *IsInitializing()* on page 357, *IsUsable()* on page 358

**IsUsable()**

Returns a Boolean value indicating if the specified attribute is usable for calculations.

**Category**

Miscellaneous

**Syntax**

```
BooleanResult = IsUsable( Attribute1, Attribute2, ... );
```

**Parameter(s)**

*Attribute1, Attribute2, ...AttributeN*

Name of one or more attributes for which you want to determine unusable quality. You can include a variable-length list of attributes.

**Return Value**

If all of the specified attributes have either Good or Uncertain quality, then true is returned. Otherwise, false is returned.

**Remarks**

To qualify as usable, the attribute must have Good or Uncertain quality. In addition, each float or double attribute cannot be a NaN (not a number).

**Examples**

```
IsUsable(TIC101.PV);  
IsUsable(TIC101.PV, PIC102.PV);
```

**See Also**

*IsBad()* on page 356, *IsGood()* on page 357, *IsInitializing()* on page 357, *IsUncertain()* on page 358

## LogCustom()

Writes a user-defined custom flag message in the Log Viewer.

### Category

Miscellaneous

### Syntax

```
LogCustom( CustomFlag, msg );
```

### Parameter

*CustomFlag*

Creates a new log flag based on the first parameter string. The first call creates the custom flag.

*msg*

The message to write to the Log Viewer. Actual string or a string attribute.

### Remarks

The log flag is disabled by default.

The message is always logged under the component "ObjectName.ScriptName". For example, "WinPlatform\_001.script1: msg", which identifies what object and what script within the object logged the error.

LogCustom() is similar to LogMessage(), but displays the message in the custom log flag when Log Custom is enabled.

The parameter help tooltip and Function Browser sample parameter list will show "LogCustom( CustomFlag, msg )" rather than "LogCustom( CustomFlag, Message )". "Message" is a reserved keyword.

### Example

```
LogCustom(EditBox1.text, "User-defined message.");
```

This statement writes to the Log Viewer as follows:

```
10/24/2005 12:49:14 PM ScriptRuntime
<ObjectName.ScriptName>: <LogFlag EditBox1> User-defined message.
```

## LogDataChangeEvent()

Logs an application change event to the Galaxy Historian.

---

Note: The LogDataChangeEvent() function works only in object scripts, not in symbol scripts.

---

### Category

Miscellaneous

### Syntax

```
LogDataChangeEvent(AttributeName, Description, OldValue, NewValue, TimeStamp);
```

### Parameters

*AttributeName*

Attribute name as a tag name.

*Description*

Description of the object.

*OldValue*

Old value of the attribute.

*NewValue*

New value of the attribute.

*TimeStamp*

The time stamp associated with the logged event. The timestamp can be UTC or local time. The TimeStamp parameter is optional. The timestamp of the logged event defaults to Now() if a TimeStamp parameter is not included.

**Remarks**

A symbol script still compiles if the LogDataChangeEvent() function is included. However, a warning message is written to the log at run time that the function is inoperable.

**Example**

This example logs an event when a pump starts or stops with a timestamp of the current time when the event occurred.

```
LogDataChangeEvent(TC104.pumpstate, "Pump04", OldState, NewState);
```

## LogError()

Writes a user-defined error message in the Log Viewer with a red error log flag.

**Category**

Miscellaneous

**Syntax**

```
LogError( msg );
```

**Parameter***msg*

The message to write to the Log Viewer. Actual string or a string attribute.

**Remarks**

The log flag is enabled by default.

The message is always logged under the component "ObjectName.ScriptName". For example, "WinPlatform\_001.script1: msg", which identifies what object and what script within the object logged the error.

LogError() is similar to LogMessage(), but displays the message in red.

The parameter help tooltip and Function Browser sample parameter list will show "LogError( msg )" rather than "LogError( Message)". "Message" is a reserved keyword.

**Example**

```
LogError("User-defined error message.");
```

This statement writes to the Log Viewer as follows:

```
10/24/2005 12:49:14 PM ScriptRuntime  
<ObjectName.ScriptName>: User-defined error message.
```

## LogMessage()

Writes a user-defined message to the Log Viewer.

**Category**

Miscellaneous

**Syntax**

```
LogMessage( msg );
```



**Parameter***msg*

The message to write to the Log Viewer. Actual string or a string attribute.

**Remarks**

This is a very powerful function for troubleshooting scripting. By strategically placing `LogMessage()` functions in your scripts, you can determine the order of script execution, performance of scripts, and identify the value of attributes both before they are changed and after they are affected by the script.

Each message posted to the Log Viewer is stamped with the exact date and time. The message always begins with the component "TagName.ScriptName" so you can tell what object and what script within the object posted the message to the log.

**Examples**

```
LogMessage("Report Script is Running");
```

The above statement writes the following to the Log Viewer:

```
10/24/2005 12:49:14 PM ScriptRuntime <TagName.ScriptName>:Report Script is
Running.
MyTag=MyTag + 10;
LogMessage("The Value of MyTag is " + Text(MyTag, "#"));
```

**LogTrace()**

Writes a user-defined trace message in the Log Viewer.

**Category**

Miscellaneous

**Syntax**

```
LogTrace( msg );
```

**Parameter***msg*

The message to write to the Log Viewer. Actual string or a string attribute.

**Remarks**

The log flag is disabled by default.

The message is always logged under the component "ObjectName.ScriptName". For example, "WinPlatform\_001.script1: msg", which identifies what object and what script within the object logged the error.

`LogTrace()` is similar to `LogMessage()`, but displays the message as Trace when Log Trace is enabled.

The parameter help tooltip and Function Browser sample parameter list will show "`LogTrace( msg )`" rather than "`LogTrace( Message )`". "Message" is a reserved keyword.

**Example**

```
LogTrace("User-defined trace message.");
```

This statement writes to the Log Viewer as follows:

```
10/24/2005 12:49:14 PM ScriptRuntime
<ObjectName.ScriptName>: User-defined trace message.
```

**LogWarning()**

Writes a user-defined error message in the Log Viewer with a yellow warning log flag.

**Category**

Miscellaneous

**Syntax**

```
LogWarning( msg );
```

**Parameter***msg*

The message to write to the Log Viewer. Actual string or a string attribute.

**Remarks**

The log flag is disabled by default.

The message is always logged under the component "ObjectName.ScriptName". For example, "WinPlatform\_001.script1: msg", which identifies what object and what script within the object logged the error.

LogWarning() is similar to LogMessage(), but displays the message as a yellow warning message.

The parameter help tooltip and Function Browser sample parameter list will show "LogWarning( msg )" rather than "LogWarning( Message )". "Message" is a reserved keyword.

**Example**

```
LogWarning("User-defined warning message.")
```

This statement writes to the Log Viewer as follows:

```
10/24/2005 12:49:14 PM ScriptRuntime
<ObjectName.ScriptName>: User-defined warning message.
```

**SendKeys()**

Sends keystrokes to an application. To the receiving application, the keys appear to be entered from the keyboard. You can use SendKeys() within a script to enter data or send commands to an application. Most keyboard keys can be used in a SendKeys() statement. Each key is represented by one or more characters, such as A for the letter A or {ENTER} for the Enter key.

**Category**

Miscellaneous

**Syntax**

```
SendKeys( KeySequence );
```

**Parameter***KeySequence*

Any key sequence or a string attribute.

**Remarks**

To specify more than one key, concatenate the codes for each character. For example, to specify the dollar sign (\$) key followed by a (b), enter \$b.

The following lists the valid send key codes for unique keyboard keys:

Key	Code
BACKSPACE	{BACKSPACE} or {BS}
BREAK	{BREAK}

Key	Code
CAPSLOCK	{CAPSLOCK}
DELETE	{DELETE} or {DEL}
DOWN	{DOWN}
END	{END}
ENTER	{ENTER} or tilde (~)
ESCAPE	{ESCAPE} or {ESC}
F1...F12	{F1}...{F12}
HOME	{HOME}
INSERT	{INSERT}
LEFT	{LEFT}
NUMLOCK	{NUMLOCK}
PAGE DOWN	{PGDN}
PAGE UP	{PGUP}
PRTSC	{PRTSC}
RIGHT	{RIGHT}
TAB	{TAB}
UP	{UP}
HOME	{HOME}

Special keys (SHIFT, CTRL, and ALT) have their own key codes:

Key	Code
SHIFT	+ (plus)
CTRL	^ (caret)
ALT	% (percent)

Enhancements to the Microsoft Hardware Abstraction Layer in Windows prevents the SendKeys() function from operating on some computers.

### Examples

To use two special keys together, use a second set of parentheses. The following statement holds down the CTRL key while pressing the ALT key, followed by p:

```
SendKeys ("^(%p)");
```

Commands can be preceded by the `ActivateApp()` command to direct the keystrokes to the proper application.

The following statement gives the computer focus to Calculator and sends the key combination 1234:

```
ActivateApp("Calculator");  
SendKeys("^1234");
```

## SetAttributeVT()

Sets the value and timestamp of an object attribute. For buffered values, only the last calculated value is captured for historization.

### Category

Miscellaneous

### Syntax

```
SetAttributeVT( Attribute, Value, TimeStamp );
```

### Parameter

#### *Attribute*

Name of the object attribute whose value and timestamp are modified. The specified attribute must belong to the object to which the script is attached.

#### *Value*

Value of the attribute, which can be a reference. The quality is always set to Good.

#### *TimeStamp*

TimeStamp that can be a reference, a variable, or a string interpreted as the computer's local time or UTC. The timestamp is converted internally to UTC format before the attribute's value is sent to the run-time component.

### Remarks

Interim calculated buffered values are NOT historized. Use SetAttributeVT2() if historization of interim values is needed.

TimeStamp can be set only for object attributes that support a timestamp. At compile time, the script cannot detect whether the attribute specified with the SetAttributeVT() function supports a timestamp or not. No warning is issued if the attribute does not support a timestamp.

### Example

This example sets an integer value and timestamp for an attribute that indicates pump RPM.

```
SetAttributeVT(me.PV, TC104.PumpRPM, LCLTIME);
```

## SetBad()

Sets the quality of an attribute to Bad.

### Category

Miscellaneous

### Syntax

```
SetBad( Attribute );
```

### Parameter

#### *Attribute*

The attribute for which you want to set the quality to Bad.

### Remarks

The specified attribute must be within the object to which the script is attached.

**Example**

```
SetBad (me.PV) ;
```

**See Also**

*SetGood()* on page 365, *SetInitializing()* on page 365, *SetUncertain()* on page 365

**SetGood()**

Sets the quality of an attribute to Good.

**Category**

Miscellaneous

**Syntax**

```
SetGood( Attribute );
```

**Parameter**

*Attribute*

The attribute for which you want to set the quality to Good.

**Remarks**

The specified attribute must be within the object to which the script is attached.

**Example**

```
SetGood (me.PV) ;
```

**See Also**

*SetBad()* on page 364, *SetInitializing()* on page 365, *SetUncertain()* on page 365

**SetInitializing()**

Sets the quality of an attribute to Initializing.

**Category**

Miscellaneous

**Syntax**

```
SetInitializing( Attribute );
```

**Parameter**

*Attribute*

The attribute for which you want to set the quality to Initializing.

**Remarks**

The specified attribute must be within the object to which the script is attached.

**Example**

```
SetInitializing (me.PV) ;
```

**See Also**

*SetBad()* on page 364, *SetGood()* on page 365, *SetUncertain()* on page 365

**SetUncertain()**

Sets the quality of an attribute to Uncertain.

## Category

Miscellaneous

## Syntax

```
SetUncertain( Attribute );
```

## Parameter

*Attribute*

The attribute for which you want to set the quality to Uncertain.

## Remarks

The specified attribute must be within the object to which the script is attached.

## Example

```
SetUncertain(me.PV);
```

## See Also

*SetBad()* on page 364, *SetGood()* on page 365, *SetInitializing()* on page 365

## SignedAlarmAck()

Acknowledges one or more alarms on tags or attributes, optionally requiring a signature if any of the indicated alarms falls within a designated priority range.

This function is supported only for client scripting and not object scripting.

## Category

Miscellaneous

## Syntax

```
int SignedAlarmAck(String Alarm_List,  
Boolean Signature_Reqd_for_Range,  
Integer Min_Priority,  
Integer Max_Priority,  
String Default_Ack_Comment,  
Boolean Ack_Comment_Is_Editable,  
String TitleBar_Caption,  
String Message_Caption  
);
```

## Parameters

*Alarm\_List*

The list of alarms to be acknowledged. The list must be a single text string with each alarm name separated by a space or a comma.

### Data Type

String

### Valid Range

Limit 1024 characters

### Additional Information

Can be a constant string, a reference, or an expression.

Only alarms on tags or attributes are supported.

If there is any invalid alarm in the list, then none of the alarms are acknowledged.

### Examples

Example 1:

```
"UD1.analog_001.HiHi"
```

The collection is represented as a text string, with alarms separated by blanks and/or commas.

Example 2:

```
"UD1.analog_001.HiHi UD9.x14.dev.major"
```

Example 3:

```
"UD1.analog_001.HiHi, UD9.x14.dev.major"
```

Example 4, an array of strings such as:

```
Pump1.AlarmArray[1] = "Pump1.Level.HiHi"
Pump1.AlarmArray[2] = "Pump1.Level.LoLo"
```

uses the function as follows:

```
SignedAlarmAck(Pump1.AlarmArray[ ], ...)
```

The script passes to the function the following single string:

```
"Pump1.Level.HiHi, Pump1.Level.LoLo"
```

*Signature\_Reqd\_for\_Range*

Indicates whether a signature is required for acknowledging alarms.

#### **Data Type**

Bool

#### **Additional Information**

Can be a constant, a reference, or an expression.

*Min\_Priority*

Represents the minimum priority value of the range for which the signature is required.

#### **Data Type**

Integer

#### **Valid Range**

1-999; must be less than or equal to the Max\_Priority value.

#### **Additional Information**

Can be a constant, a reference, or an expression.

*Max\_Priority*

Represents the maximum priority value of the range for which the signature is required.

#### **Data Type**

Integer

#### **Valid Range**

1-999; must be greater than or equal to the Min\_Priority value.

#### **Additional Information**

Can be a constant, a reference, or an expression.

*Default\_Ack\_Comment*

Comment to be shown in the **Acknowledge Alarms** dialog box.

#### **Data Type**

String

#### **Valid Range**

Limit 200 characters

#### **Additional Information**

Can be a constant, a reference or an expression.

If the parameter is empty, then no default comment is shown in the **Acknowledge Alarms** dialog box.

*Ack\_Comment\_Is\_Editable*

Indicates whether the run-time user can modify the acknowledgement comment.

**Data Type**

Bool

**Additional Information**

Can be a constant, a reference, or an expression.

If set to False, the **Comment** box in the **Acknowledge Alarms** dialog box is unavailable.

*TitleBar\_Caption*

Shows a title in the title bar of the **Acknowledge Alarms** dialog box.

**Data Type**

String

**Valid Range**

Limit 1024 characters

**Additional Information**

Can be a constant, a reference, or an expression.

**If the TitleBar\_Caption is empty, the default title, Acknowledge Alarms, is shown.**

*Message\_Caption*

Shows a customizable message to the run-time user in the **Acknowledge Alarms** dialog box.

**Data Type**

String

**Valid Range**

Limit 250 characters

**Additional Information**

Can be a constant, a reference, or an expression.

Use the parameter to provide more information on the alarm to the run-time user.

This message is not propagated to the event record.

## Return Values

Return values indicate success or failure status. A non-zero value indicates type of failure.

- 1 The user canceled the operation.  
The function writes a message to the Logger indicating user cancellation.
- 2 No alarms are waiting for acknowledgement.
- 0 The function is successful and the following are all true:
  - The function parameters are valid.
  - The user credentials are valid (or no credentials are needed).
  - The user did not cancel the operation.
  - Function wrote to the .AckMsg attributes of the indicated alarms.



- 1 The function failed due to any error that is not covered by the other specified return values.
- 2 One or more parameters were not coerced to the appropriate data type at run time.  
  
Example: Parameter is a reference with Boolean as the expected data type. At run time, reference is to a String data type that cannot be coerced to True or False.  
  
The function writes a message to the Logger.
- 3 The Alarm\_List parameter was not valid at run time.
  - String was null, empty or contained no attribute references.
  - Contained one or more items that were not valid attribute references.
  - Contained one or more attribute references that did not exist or did not identify valid alarm primitives.

If Alarm\_List contains a mixture of valid and invalid references, the function does nothing. The function does not attempt to operate on the valid references, and returns this error status.
- 4 The Min\_Priority or MaxPriority values do not fall within the range of 1 to 999.  
  
The function writes a message to the Logger indicating which parameter was out of range and showing the actual value.
- 5 The Min\_Priority value is greater than the Max\_Priority value.  
  
The function writes a message to the Logger identifying the problem and showing the actual values.

---

**Note:** A return value of zero does not indicate if the alarms are acknowledged, only that the function wrote to the AckMsg attributes. The alarms may not be acknowledged due to insufficient permission or if the alarms have already been acknowledged.

---

### Remarks

For more information about using the SignedAlarmAck() function, see the topic *Signature Security for Acknowledging Alarms*, under "Adding and Maintaining Symbol Scripts" in the *Creating and Managing Industrial Graphics User Guide*.

### Examples

```
Dim n as Integer;
n = SignedAlarmAck("UD1.analog_001.HiHi UD9.x14.dev.major", true, 1, 250,
"Acknowledged by script", true, "Acking Tank Alarms", "Acknowledge the tank
alarms");
```

Using an array of strings:

```
dim arr[2] as String;
arr[1] = "UD1.analog_001.HiHi";
arr[2] = "UD9.x14.dev.major";
n = SignedAlarmAck(arr[], true, 200, 500, "Acked by script", true, "Acking
Tank Alarms", "Please acknowledge the tank alarms.");
```

## SignedWrite()

Performs a write to an AutomationObject attribute that has a Secured Write or Verified Write security classification.

### Category

Miscellaneous

### Syntax

```
int SignedWrite(string Attribute,
object Value,
string ReasonDescription,
Bool Comment_Is_Editable,
Enum Comment_Enforcement,
string[] Predefined_Comment_List
);
```

Brackets [ ] indicate an array.

### Parameters

#### Attribute

The attribute to be updated.

#### Data Type

String

#### Additional Information

Can be a constant string, a reference, or an expression.

Supports bound and nested bound references.

For detailed examples of Attribute parameter uses, see the topic *Examples of Using the Attribute Parameter in the SignedWrite() Function* under "Managing Symbols" in the *Creating and Managing Industrial Graphics User Guide*.

#### Examples

Example 1:

```
"UserDefined_001.temp"
```

Example 2:

```
"Pump15" + ".valve4"
```

Example 3:

With UDO\_7 containing two string attributes, namestrA and namestrB set to the values "Tank1" and "Tank5" respectively, the following script writes to Tank1.Level or Tank5.Level according to whether strselect is "A" or "B":

```
Dim strselect As String;
Dim x As Indirect;
{ logic to set strselect to "A" or "B" }
x.BindTo ("UDO_7.namestr" + strselect);
SignedWrite(x + ".Level", 243, "Set " + x + " Level", true, 0, null);
```

#### Value

The value to be written.

#### Data Type

Object

#### Valid Range

Must match data type of the attribute being updated.

**Additional Information**

Can be a constant value, a reference, an expression, or NULL if nothing is to be entered.

*ReasonDescription*

Text that explains the purpose of the target attribute and the impact of changing it.

**Data Type**

String

**Valid Range**

Maximum of 256 characters.

**Additional Information**

Can be a constant string, a reference, or an expression.

The ReasonDescription is passed to the indicated Attribute as part of the write operation. The object also includes the user's write comment, if any. A Field Attribute description is used for the ReasonDescription parameter only if the attribute is a Field Attribute and it has a description (is not null). Otherwise, the Short Description for the corresponding ApplicationObject is used for the ReasonDescription parameter.

*Comment\_Is\_Editable*

Indicates whether user can edit the write comment.

**Data Type**

Bool

**Additional Information**

Can be a constant value, a reference, or an expression.

If set to True: The comment text box is enabled with exceptions. If Comment\_Is\_Editable is true and if the Comment\_Enforcement parameter is PredefinedOnly, the comment text box is disabled. At run time, the user can only select a comment from the predefined comment list.

If the Comment\_Enforcement parameter is not PredefinedOnly, the comment list and box are enabled. You can select a comment from the comment list and modify it in the comment box.

If the predefined list is empty, the comment list is not shown in the dialog box.

If set to False: The predefined comment list does not appear in the Secured Write or Verified Write dialog boxes. The editable comment text box is disabled.

*Comment\_Enforcement*

Contains choices of Optional, Mandatory and PredefinedOnly.

**Data Type**

Enum

**Enumerations**

Optional = 0

The run-time user can enter a comment or leave it blank.

Mandatory = 1

The run-time user must add a comment, either by selecting from the comment list or by entering a comment in the comment box.

PredefinedOnly = 2

The run-time user can select a comment from the comment list only. The comment text box is disabled.

**Additional Information**

Can be a constant, a reference, or an expression.

*Predefined\_Comment\_List*

An array of strings that can be used as predefined comments.

**Data Type**

String[]

**Valid Range**

Maximum of 20 comments, each with a maximum of 200 characters.

**Additional Information**

The array can be empty (number of elements is 0).

Can be a constant, a reference, an expression, or NULL if empty. Can reference an attribute that contains an array of strings.

If no predefined comment is entered, the predefined comment list is disabled at run time.

If Comment\_Is\_Editable is False, the predefined comment is still placed in the editable comment text box, but the user cannot modify it at run time.

**Return Values**

Return values indicate success or failure status. A non-zero value indicates type of failure.

- 0      The function returns a value of 0 (meaning success) if the following are all true:
- The function parameters were valid.
  - The write operation was successfully placed on the queue for Secured and Verified Writes.
  - If the user cancels the operation, a message is written to the Logger indicating user cancellation.
- 1      The function failed due to any error that is not covered by the other specified return values. This includes any error that is not covered by the other specified return values. If there is a failure, a specific message is logged in the Logger.
- 2      One or more parameters were not coerced to the appropriate data type at run time.
- Example: Parameter is a reference with Boolean as the expected data type. At run time, reference is to a String data type that cannot be coerced to True or False. The function returns this value and writes a message to the Logger.
- 3      The attribute parameter was not valid at run time.
- Attribute string was null, empty, or contained no attribute reference.
  - Attribute string contained an item that was not a valid attribute reference.
  - Attribute string contained an attribute reference that did not exist.
  - Attribute string contained an attribute reference that was not of the Secured Write or Verified Write security classification.
- The function writes a message to the Logger identifying the error and the invalid attribute string.

- 4 The Comment\_Enforcement parameter value was out of the range of valid enumerators.

### Remarks

The SignedWrite() function is supported only for client scripting and not for object scripting.

A return value of 0 does not indicate whether the attribute was updated, only that the function placed an entry on the queue to write to the attribute. The operator may decide to cancel the operation after the Secured Write or Verified write dialog box is presented. In this case the attribute is not updated and a message is placed in the Logger indicating that the user canceled the operation. Even if the user enters valid credentials and clicks **OK**, the attribute still might not have been updated because of inadequate permission or data coercion problems.

The SignedWrite() function supports the custom property passed as the first parameter with opened and closed quotation marks, "".

If you configure the custom property CP as shown in the following script, the function attempts to resolve CP and determine if it has a reference. If it has a reference, then the reference is retrieved and the write is performed on the reference.

```
SignedWrite("CP", value, reason, editable, enforcement, null);
```

For more information about using the SignedWrite() function, see the topic *Working with the SignedWrite() Function for Secured and Verified Writes* under "Managing Symbols" in the *Creating and Managing Industrial Graphics User Guide*.

### Examples

```
SignedWrite ("UserDefined_001.temp", 185, "This will change the oven temperature", true, 1, null);
```

The following example shows the user an array of predefined comments:

```
Dim n as Integer;
n = SignedWrite("UserDefined_001.temp", 185, "This will change the oven temperature", true, 1, UserDefined_001.OvenCommentArray[ ]);
```

where UserDefined\_001.OvenCommentArray is an attribute containing an array of strings.

## WriteStatus()

Returns the enumerated write status of the last write to the specified attribute.

### Category

Miscellaneous

### Syntax

```
Result = WriteStatus( Attribute );
```

### Parameter

*Attribute*

The attribute for which you want to return write status.

### Return Value

The return statuses are:

- MxStatusOk
- MxStatusPending
- MxStatusWarning
- MxStatusCommunicationError
- MxStatusConfigurationError

- MxStatusOperationalError
- MxStatusSecurityError
- MxStatusSoftwareError
- MxStatusOtherError

### Remarks

If the attribute has never been written to, this function returns MxStatusOk. This function always returns MxStatusOk for attributes that do not support a calculated (non-Good) quality.

### Example

```
WriteStatus(TIC101.SP);
```

## WWControl()

Restores, minimizes, maximizes, or closes an application.

### Category

Miscellaneous

### Syntax

```
WWControl( AppTitle, ControlType );
```

### Parameters

#### *AppTitle*

The name of the application title to be controlled. Actual string or a string attribute.

#### *ControlType*

Determines how the application is controlled. Possible values are shown below. These actions are identical to clicking on their corresponding selections in the application's Control Menu. Actual string or a string attribute.

"Restore" = Activates and shows the application's window.

"Minimize" = Activates a window and shows it as an icon.

"Maximize" = Activates and shows the application's window.

"Close" = Closes an application.

### Example

```
WWControl("Calculator", "Restore");
```

### See Also

*ActivateApp()* on page 355

## String Functions

Use string functions to work with character strings and string values.

### DText()

Returns one of two possible strings, depending on the value of the *Discrete* parameter.

### Category

String

### Syntax

```
StringResult = DText( Discrete, OnMsg, OffMsg );
```

## Parameters

*Discrete*

A Boolean value or Boolean attribute.

*OnMsg*

The message that is shown when the value of *Discrete* equals true.

*OffMsg*

The message shown when *Discrete* equals false.

## Example

```
StringResult = DText(me.temp > 150, "Too hot", "Just right");
```

## StringASCII()

Returns the ASCII value of the first character in a specified string.

### Category

String

### Syntax

```
IntegerResult = StringASCII( Char );
```

### Parameter

*Char*

Alphanumeric character or string or string attribute.

### Remarks

When this function is processed, only the single character is tested or affected. If the string provided to `StringASCII` contains more than one character, only the first character of the string is tested.

### Examples

```
StringASCII("A"); ' returns 65;  
StringASCII("A Mixer is Running"); ' returns 65;  
StringASCII("a mixer is running"); ' returns 97;
```

### See Also

[StringChar\(\)](#) on page 375, [StringFromIntg\(\)](#) on page 378, [StringFromReal\(\)](#) on page 378, [StringFromTime\(\)](#) on page 379, [StringInString\(\)](#) on page 381, [StringLeft\(\)](#) on page 381, [StringLen\(\)](#) on page 382, [StringLower\(\)](#) on page 383, [StringMid\(\)](#) on page 383, [StringReplace\(\)](#) on page 384, [StringRight\(\)](#) on page 385, [StringSpace\(\)](#) on page 385, [StringTest\(\)](#) on page 386, [StringToIntg\(\)](#) on page 387, [StringToReal\(\)](#) on page 387, [StringTrim\(\)](#) on page 388, [StringUpper\(\)](#) on page 389, [Text\(\)](#) on page 389

## StringChar()

Returns the character corresponding to a specified ASCII code.

### Category

String

### Syntax

```
StringResult = StringChar( ASCII );
```

### Parameter

*ASCII*

ASCII code or an integer attribute.

**Remarks**

Use the `StringChar` function to add ASCII characters not normally represented on the keyboard to a string attribute.

This function is also useful for SQL commands. The where expression sometimes requires double quotation marks around string values, so use `StringChar(34)`.

**Example**

In this example, a [Carriage Return (13)] and [Line Feed (10)] are added to the end of `StringAttribute` and passed to `ControlString`. Inserting characters out of the normal 32-126 range of displayable ASCII characters can be very useful for creating control codes for external devices such as printers or modems.

```
ControlString = StringAttribute+StringChar(13)+StringChar(10);
```

**StringCompare()**

Compares a string value with another string.

**Category**

String

**Syntax**

```
StringCompare( Text1, Text2 );
```

**Parameters**

*Text1*

First string in the comparison.

*Text2*

Second string in the comparison.

**Return Value**

The return value is zero if the strings are identical, -1 if *Text1*'s value is less than *Text2*, or 1 if *Text1*'s value is greater than *Text2*.

**Example**

```
Result = StringCompare ("Text1","Text2"); (or)
```

```
Result = StringCompare (MText1,MText2);
```

Where *Result* is an Integer or Real tag and *MText1* and *MText2* are Memory Message tags.

**See Also**

*StringASCII()* on page 375, *StringChar()* on page 375, *StringFromIntg()* on page 378, *StringFromReal()* on page 378, *StringFromTime()* on page 379, *StringFromTimeLocal()* on page 380, *StringInString()* on page 381, *StringLeft()* on page 381, *StringLen()* on page 382, *StringLower()* on page 383, *StringMid()* on page 383, *StringReplace()* on page 384, *StringRight()* on page 385, *StringSpace()* on page 385, *StringTest()* on page 386, *StringToIntg()* on page 387, *StringToReal()* on page 387, *StringTrim()* on page 388, *StringUpper()* on page 389, *Text()* on page 389

**StringCompareNoCase()**

Compares a string value with another string and ignores the case.

**Category**

String

**Syntax**

```
SStringCompareNoCase( Text1, Text2 );
```



**Parameters***Text1*

First string in the comparison.

*Text2*

Second string in the comparison.

**Return Value**

The return value is zero if the strings are identical (ignoring case), -1 if Text1's value is less than Text2 (ignoring case), or 1 if Text1's value is greater than Text2 (ignoring case).

**Example**

```
Result = StringCompareNoCase ("Text1", "TEXT1"); (or)
```

```
Result = StringCompareNoCase (MText1, MText2);
```

Where Result is an Integer or Real tag and MText1 and MText2 are Memory Message tags.

**See Also**

*StringASCII()* on page 375, *StringChar()* on page 375, *StringFromIntg()* on page 378, *StringFromReal()* on page 378, *StringFromTime()* on page 379, *StringFromTimeLocal()* on page 380, *StringInString()* on page 381, *StringLeft()* on page 381, *StringLen()* on page 382, *StringLower()* on page 383, *StringMid()* on page 383, *StringReplace()* on page 384, *StringRight()* on page 385, *StringSpace()* on page 385, *StringTest()* on page 386, *StringToIntg()* on page 387, *StringToReal()* on page 387, *StringTrim()* on page 388, *StringUpper()* on page 389, *Text()* on page 389

**StringFromGMTTimeToLocal()**

Converts a time value (in seconds since Jan-01-1970) to a particular string representation. This is the same as *StringFromTime()*.

**Category**

String

**Syntax**

```
MessageResult=StringFromGMTTimeToLocal (SecsSince1-1-70, StringType) ;
```

**Parameters***SecsSince1-1-70*

Is converted to the StringType specified and the result is stored in MessageResult.

*StringType*

Determines the display method:

1 = Displays the date in the same format set from the windows control Panel. (Similar to that displayed for \$DateString.)

2 = Displays the time in the same format set from the Windows control Panel. (Similar to that displayed for \$TimeString.)

3 = Displays a 24-character string indicating both the date and time: "Wed Jan 02 02:03:55 1993"

4 = Displays the short form for the day of the week: "Wed"

5 = Displays the long form for the day of the week: "Wednesday"

**Remarks**

Any adjustments necessary due to Daylight Savings Time are automatically applied to the return result. Therefore, it is not necessary to make any manual adjustments to the input value to convert to DST.

## Example

This example assumes that the time zone on the local node is Pacific Standard Time (UTC-0800). The UTC time passed to the function is 12:00:00 AM on Friday, 1/2/1970. Since PST is 8 hours behind UTC, the function will return the following results:

```
StringFromGMTTimeToLocal(86400, 1); ' returns "1/1/1970"
StringFromGMTTimeToLocal(86400, 2); ' returns "04:00:00 PM"
StringFromGMTTimeToLocal(86400, 3); ' returns "Thu Jan 01 16:00:00 1970"
StringFromGMTTimeToLocal(86400, 4); ' returns "Thu"
StringFromGMTTimeToLocal(86400, 5); ' returns "Thursday"
```

## See Also

*StringASCII()* on page 375, *StringChar()* on page 375, *StringFromIntg()* on page 378, *StringFromReal()* on page 378, *StringFromTime()* on page 379, *StringFromTimeLocal()* on page 380, *StringInString()* on page 381, *StringLeft()* on page 381, *StringLen()* on page 382, *StringLower()* on page 383, *StringMid()* on page 383, *StringReplace()* on page 384, *StringRight()* on page 385, *StringSpace()* on page 385, *StringTest()* on page 386, *StringToIntg()* on page 387, *StringToReal()* on page 387, *StringTrim()* on page 388, *StringUpper()* on page 389, *Text()* on page 389

## StringFromIntg()

Converts an integer value into its string representation in another base and returns the result.

### Category

String

### Syntax

```
StringResult = StringFromIntg( Number, numberBase );
```

### Parameters

*Number*

Number to convert. Any number or an integer attribute.

*numberBase*

Base to use in conversion. Any number or an integer attribute.

### Examples

```
StringFromIntg(26, 2); ' returns "11010"
StringFromIntg(26, 8); ' returns "32"
StringFromIntg(26, 16); ' returns "1A"
```

### See Also

*StringASCII()* on page 375, *StringChar()* on page 375, *StringFromReal()* on page 378, *StringFromTime()* on page 379, *StringInString()* on page 381, *StringLeft()* on page 381, *StringLen()* on page 382, *StringLower()* on page 383, *StringMid()* on page 383, *StringReplace()* on page 384, *StringRight()* on page 385, *StringSpace()* on page 385, *StringTest()* on page 386, *StringToIntg()* on page 387, *StringToReal()* on page 387, *StringTrim()* on page 388, *StringUpper()* on page 389, *Text()* on page 389

## StringFromReal()

Converts a real value into its string representation, either as a floating-point number or in exponential notation, and returns the result.

### Category

String

### Syntax

```
StringResult = StringFromReal( Number, Precision, Type );
```

## Parameters

### *Number*

Converted to the *Precision* and *Type* specified. Any number or a float attribute.

### *Precision*

Specifies how many decimal places is shown. Any number or an integer attribute.

### *Type*

A string value that determines the display method. Possible values are:

f = Display in floating-point notation.

e = Display in exponential notation with a lowercase "e."

E = Display in exponential notation with an uppercase "E" followed by a plus sign and at least three exponential digits.

## Examples

```
StringFromReal (263.355, 2, "f"); ' returns "263.36";
StringFromReal (263.355, 2, "e"); ' returns "2.63e2";
StringFromReal (263.355, 2, "E"); ' returns "2.63 E+002";
```

## See Also

*StringASCII()* on page 375, *StringChar()* on page 375, *StringFromIntg()* on page 378, *StringFromTime()* on page 379, *StringInString()* on page 381, *StringLeft()* on page 381, *StringLen()* on page 382, *StringLower()* on page 383, *StringMid()* on page 383, *StringReplace()* on page 384, *StringRight()* on page 385, *StringSpace()* on page 385, *StringTest()* on page 386, *StringToIntg()* on page 387, *StringToReal()* on page 387, *StringTrim()* on page 388, *StringUpper()* on page 389, *Text()* on page 389

## StringFromTime()

Converts a time value (in seconds since January 1, 1970) into a particular string representation and returns the result.

## Category

String

## Syntax

```
StringResult = StringFromTime ( SecsSince1-1-70, StringType );
```

## Parameters

### *SecsSince1-1-70*

Converted to the *StringType* specified.

### *StringType*

Determines the display method. Possible values are:

1 = Shows the date in the same format set from the Windows Control Panel.

2 = Shows the time in the same format set from the Windows Control Panel.

3 = Shows a 24-character string indicating both the date and time: "Wed Jan 02 02:03:55 1993"

4 = Shows the short form for a day of the week: "Wed"

5 = Shows the long form for a day of the week: "Wednesday"

## Remarks

The time value is UTC equivalent: number of elapsed seconds since January 1, 1970 GMT. The value returned reflects the local time.

**Examples**

```
StringFromTime(86400, 1); ' returns "1/2/1970"
StringFromTime(86400, 2); ' returns "12:00:00 AM"
StringFromTime(86400, 3); ' returns "Fri Jan 02 00:00:00 1970"
StringFromTime(86400, 4); ' returns "Fri"
StringFromTime(86400, 5); ' returns "Friday"
```

**See Also**

*StringASCII()* on page 375, *StringChar()* on page 375, *StringFromIntg()* on page 378, *StringFromReal()* on page 378, *StringFromTime()* on page 379, *StringInString()* on page 381, *StringLeft()* on page 381, *StringLen()* on page 382, *StringLower()* on page 383, *StringMid()* on page 383, *StringReplace()* on page 384, *StringRight()* on page 385, *StringSpace()* on page 385, *StringTest()* on page 386, *StringToIntg()* on page 387, *StringToReal()* on page 387, *StringTrim()* on page 388, *StringUpper()* on page 389, *Text()* on page 389

**StringFromTimeLocal()**

Converts a time value (in seconds since Jan-01-1970) into a particular string representation. The value returned also represents local time.

**Category**

String

**Syntax**

```
MessageResult=StringFromTimeLocal(SecsSince1-1-70,
```

```
StringType);
```

**Parameters**

*SecsSince1-1-70*

Is converted to the *StringType* specified and the result is stored in *MessageResult*.

*StringType*

Determines the display method:

1 = Displays the date in the same format set from the windows control Panel. (Similar to that displayed for \$DateString.)

2 = Displays the time in the same format set from the Windows control Panel. (Similar to that displayed for \$TimeString.)

3 = Displays a 24-character string indicating both the date and time: "Wed Jan 02 02:03:55 1993"

4 = Displays the short form for the day of the week: "Wed"

5 = Displays the long form for the day of the week: "Wednesday"

**Remarks**

Any adjustments necessary due to Daylight Savings Time will automatically be applied to the return result. Therefore, it is not necessary to make any manual adjustments for DST to the input value.

**Example**

```
StringFromTimeLocal(86400, 1); ' returns "1/2/1970"
StringFromTimeLocal(86400, 2); ' returns "12:00:00 AM"
StringFromTimeLocal(86400, 3); ' returns "Fri Jan 02 00:00:00 1970"
StringFromTimeLocal(86400, 4); ' returns "Fri"
StringFromTimeLocal(86400, 5); ' returns "Friday"
```

**See Also**

*StringASCII()* on page 375, *StringChar()* on page 375, *StringFromIntg()* on page 378, *StringFromReal()* on page 378, *StringFromTime()* on page 379, *StringInString()* on page 381, *StringLeft()* on page 381, *StringLen()* on page 382, *StringLower()* on page 383, *StringMid()* on page 383, *StringReplace()* on page 384, *StringRight()* on page 385, *StringSpace()* on page 385, *StringTest()* on page 386, *StringToIntg()* on page 387, *StringToReal()* on page 387, *StringTrim()* on page 388, *StringUpper()* on page 389, *Text()* on page 389

**StringInString()**

Returns the position in a string of text where a specified string first occurs.

**Category**

String

**Syntax**

```
IntegerResult = StringInString( Text, SearchFor, StartPos, CaseSens );
```

**Parameters***Text*

The string that is searched. Actual string or a string attribute.

*SearchFor*

The string to be searched for. Actual string or a string attribute.

*StartPos*

Determines the position in the text where the search begins. Any number or an integer attribute.

*CaseSens*

Determines whether the search is case-sensitive.

0 = Not case-sensitive

1 = Case-sensitive

Any number or an integer attribute.

**Remarks**

If multiple occurrences of *SearchFor* are found, the location of the first is returned.

**Examples**

```
StringInString("The mixer is running", "mix", 1, 0) ' returns 5;
StringInString("Today is Thursday", "day", 1, 0) ' returns 3;
StringInString("Today is Thursday", "day", 10, 0) ' returns 15;
StringInString("Today is Veteran's Day", "Day", 1, 1) ' returns 20;
StringInString("Today is Veteran's Day", "Night", 1, 1) ' returns 0;
```

**See Also**

*StringASCII()* on page 375, *StringChar()* on page 375, *StringFromIntg()* on page 378, *StringFromReal()* on page 378, *StringFromTime()* on page 379, *StringLeft()* on page 381, *StringLen()* on page 382, *StringLower()* on page 383, *StringMid()* on page 383, *StringReplace()* on page 384, *StringRight()* on page 385, *StringSpace()* on page 385, *StringTest()* on page 386, *StringToIntg()* on page 387, *StringToReal()* on page 387, *StringTrim()* on page 388, *StringUpper()* on page 389, *Text()* on page 389

**StringLeft()**

Returns a specified number of characters in a string value, starting with the leftmost string character.

## Category

String

## Syntax

```
StringResult = StringLeft( Text, Chars );
```

## Parameters

*Text*

Actual string or a string attribute.

*Chars*

Number of characters to return or an integer attribute.

## Remarks

If *Chars* is set to 0, the entire string is returned.

## Examples

```
StringLeft("The Control Pump is On", 3) ' returns "The";  
StringLeft("Pump 01 is On", 4) ' returns "Pump";  
StringLeft("Pump 01 is On", 96) ' returns "Pump 01 is On";  
StringLeft("The Control Pump is On", 0) ' returns "The Control Pump is On";
```

## See Also

*StringASCII()* on page 375, *StringChar()* on page 375, *StringFromIntg()* on page 378, *StringFromReal()* on page 378, *StringFromTime()* on page 379, *StringInString()* on page 381, *StringLen()* on page 382, *StringLower()* on page 383, *StringMid()* on page 383, *StringReplace()* on page 384, *StringRight()* on page 385, *StringSpace()* on page 385, *StringTest()* on page 386, *StringToIntg()* on page 387, *StringToReal()* on page 387, *StringTrim()* on page 388, *StringUpper()* on page 389, *Text()* on page 389

## StringLen()

Returns the number of characters in a string.

## Category

String

## Syntax

```
IntegerResult = StringLen( Text );
```

## Parameter

*Text*

Actual string or a string attribute.

## Remarks

All the characters in the string attribute are counted, including blank spaces and those not normally shown on the screen.

## Examples

```
StringLen("Twelve percent") ' returns 14;  
StringLen("12%") ' returns 3;  
StringLen("The end." + StringChar(13)) ' returns 9;
```

The carriage return character is ASCII 13.

**See Also**

*StringASCII()* on page 375, *StringChar()* on page 375, *StringFromIntg()* on page 378, *StringFromReal()* on page 378, *StringFromTime()* on page 379, *StringInString()* on page 381, *StringLeft()* on page 381, *StringLower()* on page 383, *StringMid()* on page 383, *StringReplace()* on page 384, *StringRight()* on page 385, *StringSpace()* on page 385, *StringTest()* on page 386, *StringToIntg()* on page 387, *StringToReal()* on page 387, *StringTrim()* on page 388, *StringUpper()* on page 389, *Text()* on page 389

**StringLower()**

Converts all uppercase characters in text string to lowercase and returns the result.

**Category**

String

**Syntax**

```
StringResult = StringLower( Text );
```

**Parameter**

*Text*

String to be converted to lowercase. Actual string or a string attribute.

**Remarks**

Lowercase characters, symbols, numbers, and other special characters are not affected.

**Examples**

```
StringLower("TURBINE") ' returns "turbine";
StringLower("22.2 Is The Value") ' returns "22.2 is the value";
```

**See Also**

*StringASCII()* on page 375, *StringChar()* on page 375, *StringFromIntg()* on page 378, *StringFromReal()* on page 378, *StringFromTime()* on page 379, *StringInString()* on page 381, *StringLeft()* on page 381, *StringLen()* on page 382, *StringMid()* on page 383, *StringReplace()* on page 384, *StringRight()* on page 385, *StringSpace()* on page 385, *StringTest()* on page 386, *StringToIntg()* on page 387, *StringToReal()* on page 387, *StringTrim()* on page 388, *StringUpper()* on page 389, *Text()* on page 389

**StringMid()**

Extracts a specific number of characters from a starting point within a string and returns the extracted character string as the result.

**Category**

String

**Syntax**

```
StringResult = StringMid( Text, StartChar, Chars );
```

**Parameters**

*Text*

Actual string or a string attribute to extract a range of characters.

*StartChar*

The position of the first character within the string to extract. Any number or an integer attribute.

*Chars*

The number of characters within the string to return. Any number or an integer attribute.

## Remarks

This function is slightly different than the *StringLeft()* on page 381 function and *StringRight()* on page 385 function in that it allows you to specify both the start and end of the string that is to be extracted.

## Examples

```
StringMid("The Furnace is Overheating",5,7); ' returns "Furnace";
StringMid("The Furnace is Overheating",13,3); ' returns "is ";
StringMid("The Furnace is Overheating",16,50); ' returns "Overheating"
```

## See Also

*StringASCII()* on page 375, *StringChar()* on page 375, *StringFromIntg()* on page 378, *StringFromReal()* on page 378, *StringFromTime()* on page 379, *StringInString()* on page 381, *StringLeft()* on page 381, *StringLen()* on page 382, *StringLower()* on page 383, *StringReplace()* on page 384, *StringRight()* on page 385, *StringSpace()* on page 385, *StringTest()* on page 386, *StringToIntg()* on page 387, *StringToReal()* on page 387, *StringTrim()* on page 388, *StringUpper()* on page 389, *Text()* on page 389

## StringReplace()

Replaces or changes specific parts of a provided string and returns the result.

### Category

String

### Syntax

```
StringResult = StringReplace( Text, SearchFor, ReplaceWith, CaseSens,
NumToReplace, MatchWholeWords );
```

### Parameters

#### *Text*

The string in which characters, words, or phrases will be replaced. Actual string or a string attribute.

#### SearchFor

The string to search for and replace. Actual string or a string attribute.

#### *ReplaceWith*

The replacement string. Actual string or a string attribute.

#### *CaseSens*

Determines whether the search is case-sensitive. (0=no and 1=yes) Any number or an integer attribute.

#### *NumToReplace*

Determines the number of occurrences to replace. Any number or an integer attribute. To indicate all occurrences, set this value to -1.

#### *MatchWholeWords*

Determines whether the function limits its replacement to whole words. (0=no and 1=yes) Any number or an integer attribute. If *MatchWholeWords* is turned on (set to 1) and the *SearchFor* is set to "and", the "and" in "handle" are not replaced. If the *MatchWholeWords* is turned off (set to 0), it is replaced.

## Remarks

Use this function to replace characters, words, or phrases within a string.

The *StringReplace()* on page 384 function does not recognize special characters, such as @ # \$ % & \* ( ). It reads them as delimiters. For example, if the function *StringReplace()* on page 384 (abc#,abc#,1234,0,1,1) is processed, there is no replacement. The # sign is read as a delimiter instead of a character.



**Examples**

```
StringReplace("In From Within", "In", "Out", 0, 1, 0) ' returns "Out From Within"
(replaces only the first one);
StringReplace("In From Within", "In", "Out", 0, -1, 0) ' returns "Out From without"
(replaces all occurrences);
StringReplace("In From Within", "In", "Out", 1, -1, 0) ' returns "Out From Within"
(replaces all that match case);
StringReplace("In From Within", "In", "Out", 0, -1, 1) ' returns "Out From Within"
(replaces all that are whole words);
```

**See Also**

*StringASCII()* on page 375, *StringChar()* on page 375, *StringFromIntg()* on page 378, *StringFromReal()* on page 378, *StringFromTime()* on page 379, *StringInString()* on page 381, *StringLeft()* on page 381, *StringLen()* on page 382, *StringLower()* on page 383, *StringMid()* on page 383, *StringRight()* on page 385, *StringSpace()* on page 385, *StringTest()* on page 386, *StringToIntg()* on page 387, *StringToReal()* on page 387, *StringTrim()* on page 388, *StringUpper()* on page 389, *Text()* on page 389

**StringRight()**

Returns the specified number of characters starting at the right-most character of text.

**Category**

String

**Syntax**

```
StringResult = StringRight( Text, Chars );
```

**Parameters**

*Text*

Actual string or a string attribute.

*Chars*

The number of characters to return or an integer attribute.

**Remarks**

If *Chars* is set to 0, the entire string is returned.

**Examples**

```
StringRight("The Pump is On", 2) ' returns "On";
StringRight("The Pump is On", 5) ' returns "is On";
StringRight("The Pump is On", 87) ' returns "The Pump is On";
StringRight("The Pump is On", 0) ' returns "The Pump is On";
```

**See Also**

*StringASCII()* on page 375, *StringChar()* on page 375, *StringFromIntg()* on page 378, *StringFromReal()* on page 378, *StringFromTime()* on page 379, *StringInString()* on page 381, *StringLeft()* on page 381, *StringLen()* on page 382, *StringLower()* on page 383, *StringMid()* on page 383, *StringReplace()* on page 384, *StringSpace()* on page 385, *StringTest()* on page 386, *StringToIntg()* on page 387, *StringToReal()* on page 387, *StringTrim()* on page 388, *StringUpper()* on page 389, *Text()* on page 389

**StringSpace()**

Generates a string of spaces either within a string attribute or within an expression and returns the result.

**Category**

String

**Syntax**

```
StringResult = StringSpace( NumSpaces );
```

**Parameter**

*NumSpaces*

Number of spaces to return. Any number or an integer attribute.

**Examples**

All spaces are represented by the "x" character:

```
StringSpace(4) ' returns "xxxx";
```

```
"Pump" + StringSpace(1) + "Station" ' returns "PumpxStation";
```

**See Also**

*StringASCII()* on page 375, *StringChar()* on page 375, *StringFromIntg()* on page 378, *StringFromReal()* on page 378, *StringFromTime()* on page 379, *StringInString()* on page 381, *StringLeft()* on page 381, *StringLen()* on page 382, *StringLower()* on page 383, *StringMid()* on page 383, *StringReplace()* on page 384, *StringRight()* on page 385, *StringTest()* on page 386, *StringToIntg()* on page 387, *StringToReal()* on page 387, *StringTrim()* on page 388, *StringUpper()* on page 389, *Text()* on page 389

**StringTest()**

Tests the first character of text to determine whether it is of a certain type and returns the result.

**Category**

String

**Syntax**

```
DiscreteResult = StringTest( Text, TestType );
```

**Parameters**

*Text*

String that function acts on. Actual string or a string attribute.

*TestType*

Determines the type of test. Possible values are:

1 = Alphanumeric character ('A-Z', 'a-z' and '0-9')

2 = Numeric character ('0-9')

3 = Alphabetic character ('A-Z' and 'a-z')

4 = Uppercase character ('A-Z')

5 = Lowercase character ('a-z')

6 = Punctuation character (0x21-0x2F)

7 = ASCII characters (0x00 - 0x7F)

8 = Hexadecimal characters ('A-F' or 'a-f' or '0-9')

9 = Printable character (0x20-0x7E)

10 = Control character (0x00-0x1F or 0x7F)

11 = White Space characters (0x09-0x0D or 0x20)

**Remarks**

*StringTest()* on page 386 function returns true to *DiscreteResult* if the first character in *Text* is of the type specified by *TestType*. Otherwise, false is returned. If the *StringTest()* on page 386 function contains more than one character, only the first character of the attribute is tested.

**Examples**

```
StringTest("ACB123",1) ' returns 1;
StringTest("ABC123",5) ' returns 0;
```

**See Also**

*StringASCII()* on page 375, *StringChar()* on page 375, *StringFromIntg()* on page 378, *StringFromReal()* on page 378, *StringFromTime()* on page 379, *StringInString()* on page 381, *StringLeft()* on page 381, *StringLen()* on page 382, *StringLower()* on page 383, *StringMid()* on page 383, *StringReplace()* on page 384, *StringRight()* on page 385, *StringSpace()* on page 385, *StringToIntg()* on page 387, *StringToReal()* on page 387, *StringTrim()* on page 388, *StringUpper()* on page 389, *Text()* on page 389

**StringToIntg()**

Converts the numeric value of a string to an integer value and returns the result.

**Category**

String

**Syntax**

```
IntegerResult = StringToIntg( Text );
```

**Parameter**

*Text*

String that function acts on. Actual string or a string attribute.

**Remarks**

When this statement is evaluated, the system reads the first character of the string for a numeric value. If the first character is other than a number, the string's value is equated to zero (0). Blank spaces are ignored. If the first character is a number, the system continues to read the subsequent characters until a non-numeric value is detected.

**Examples**

```
StringToIntg("ABCD"); ' returns 0;
StringToIntg("22.2 is the Value"); ' returns 22 (since integers are whole
numbers);
StringToIntg("The Value is 22"); ' returns 0;
```

**See Also**

*StringASCII()* on page 375, *StringChar()* on page 375, *StringFromIntg()* on page 378, *StringFromReal()* on page 378, *StringFromTime()* on page 379, *StringInString()* on page 381, *StringLeft()* on page 381, *StringLen()* on page 382, *StringLower()* on page 383, *StringMid()* on page 383, *StringReplace()* on page 384, *StringRight()* on page 385, *StringSpace()* on page 385, *StringTest()* on page 386, *StringToReal()* on page 387, *StringTrim()* on page 388, *StringUpper()* on page 389, *Text()* on page 389

**StringToReal()**

Converts the numeric value of a string to a real (floating point) value and returns the result.

**Category**

String

**Syntax**

```
RealResult = StringToReal( Text );
```

**Parameter**

*Text*

String that function acts on. Actual string or a string attribute.

## Remarks

When this statement is evaluated, the system reads the first character of the string for a numeric value. If the first character is other than a number (blank spaces are ignored), the string's value is equated to zero (0). If the first character is found to be a number, the system continues to read the subsequent characters until a non-numeric value is encountered.

## Examples

```
StringToReal("ABCD"); ' returns 0;
StringToReal("22.261 is the value"); ' returns 22.261;
StringToReal("The Value is 2"); ' returns 0;
```

## See Also

*StringASCII()* on page 375, *StringChar()* on page 375, *StringFromIntg()* on page 378, *StringFromReal()* on page 378, *StringFromTime()* on page 379, *StringInString()* on page 381, *StringLeft()* on page 381, *StringLen()* on page 382, *StringLower()* on page 383, *StringMid()* on page 383, *StringReplace()* on page 384, *StringRight()* on page 385, *StringSpace()* on page 385, *StringTest()* on page 386, *StringToIntg()* on page 387, *StringTrim()* on page 388, *StringUpper()* on page 389, *Text()* on page 389

## StringTrim()

Removes unwanted spaces from text and returns the result.

### Category

String

### Syntax

```
StringResult = StringTrim( Text, TrimType );
```

### Parameter

#### *Text*

String that is trimmed of spaces. Actual string or a string attribute.

#### *TrimType*

Determines how the string is trimmed. Possible values are:

- 1 = Remove leading spaces to the left of the first non-space character
- 2 = Remove trailing spaces to the right of the last non-space character
- 3 = Remove all spaces except for single spaces between words

## Remarks

The text is searched for white-spaces (ASCII 0x09-0x0D or 0x20) that are to be removed. *TrimType* determines the method used by the function:

## Examples

All spaces are represented by the "x" character.

```
StringTrim("xxxxxThisxisaxxtestxxxxx", 1) ' returns "Thisxisaxxtestxxxxx";
StringTrim("xxxxxThisxisaxxtestxxxxx", 2) ' returns "xxxxxThisxisaxxtest";
StringTrim("xxxxxThisxisaxxtestxxxxx", 3) ' returns "Thisxisaxxtest";
```

The *StringReplace()* on page 384 function can remove ALL spaces from a specified a string attribute. Simply replace all the space characters with a "null."

**See Also**

*StringASCII()* on page 375, *StringChar()* on page 375, *StringFromIntg()* on page 378, *StringFromReal()* on page 378, *StringFromTime()* on page 379, *StringInString()* on page 381, *StringLeft()* on page 381, *StringLen()* on page 382, *StringLower()* on page 383, *StringMid()* on page 383, *StringReplace()* on page 384, *StringRight()* on page 385, *StringSpace()* on page 385, *StringTest()* on page 386, *StringToIntg()* on page 387, *StringToReal()* on page 387, *StringUpper()* on page 389, *Text()* on page 389

**StringUpper()**

Converts all lowercase text characters to uppercase and returns the result.

**Category**

String

**Syntax**

```
StringResult = StringUpper( Text );
```

**Parameter**

*Text*

String to be converted to uppercase. Actual string or a string attribute.

**Remarks**

Uppercase characters, symbols, numbers, and other special characters are not affected.

**Examples**

```
StringUpper("abcd"); ' returns "ABCD";
StringUpper("22.2 is the value"); ' returns "22.2 IS THE VALUE";
```

**See Also**

*StringASCII()* on page 375, *StringChar()* on page 375, *StringFromIntg()* on page 378, *StringFromReal()* on page 378, *StringFromTime()* on page 379, *StringInString()* on page 381, *StringLeft()* on page 381, *StringLen()* on page 382, *StringLower()* on page 383, *StringMid()* on page 383, *StringReplace()* on page 384, *StringRight()* on page 385, *StringSpace()* on page 385, *StringTest()* on page 386, *StringToIntg()* on page 387, *StringToReal()* on page 387, *StringTrim()* on page 388, *Text()* on page 389

**Text()**

Converts a number to text based on a specified format.

**Category**

String

**Syntax**

```
StringResult = Text( Number, Format );
```

**Parameters**

*Number*

Any number or numeric attribute.

*Format*

Format to use in conversion. Actual string or a string attribute.

**Examples**

```
Text(66,"#.00"); ' returns 66.00;
Text(22.269,"#.00"); ' returns 22.27;
Text(9.999,"#.00"); ' returns 10.00;
```

The following example shows how to use this function within another function:

```
LogMessage("The current value of FreezerRoomTemp is:" + Text (FreezerRoomTemp,
"#.#"));
```

In the following example, MessageTag is set to "One=1 Two=2".

```
MessageTag = "One + " + Text(1, "#") + StringChar(32) + "Two +" + Text(2, "#");
```

### See Also

*StringFromIntg()* on page 378, *StringToIntg()* on page 387, *StringFromReal()* on page 378, *StringToReal()* on page 387

## WWStringFromTime()

Converts a time value given in local time into UTC time (Coordinated Universal Time), and displays the result as a string.

### Category

String

### Syntax

```
MessageResult = wwStringFromTime (SecsSince1-1-70, StringType);
```

### Parameters

*SecsSince1-1-70*

Integer Type. Number of Seconds elapsed since Jan 01 00:00:00 1970.

*StringType*

Determines the display method:

1 = Displays the date in the same format set from the windows control Panel. (Similar to that displayed for \$DateString.)

2 = Displays the time in the same format set from the Windows control Panel. (Similar to that displayed for \$TimeString.)

3 = Displays a 24-character string indicating both the date and time: "Wed Jan 02 02:03:55 1993"

4 = Displays the short form for the day of the week: "Wed"

5 = Displays the long form for the day of the week: "Wednesday"

### Remarks

Any adjustments necessary due to Daylight Savings Time will automatically be applied to the return result. Therefore, it is not necessary to make any manual adjustments for DST to the input value.

### Example

This example assumes that the time zone on the local node is Pacific Standard Time (UTC-0800). The local time passed to the function is 04:00:00 PM on Thursday, 1/1/1970. Since PST is 8 hours behind UTC, the function will return the following results:

```
wwStringFromTime (57600, 1) will return "1/2/70"
wwStringFromTime (57600, 2) will return "12:00:00 AM"
wwStringFromTime (57600, 3) will return "Fri Jan 02 00:00:00 1970"
wwStringFromTime (57600, 4) will return "Fri"
wwStringFromTime (57600, 5) will return "Friday"
```

## System Functions

Use system functions to interact with the operating system or other core system functions, such as ActiveX objects.

## CreateObject()

Creates an ActiveX (COM) object.

### Category

System

### Syntax

```
ObjectResult = CreateObject ( ProgID );
```

### Parameter

*ProgID*

The program ID (as a string) of the object to be created.

### Example

```
CreateObject ("ADODB.Connection");
```

## Now()

Returns the current time.

### Category

System

### Syntax

```
TimeValue = Now();
```

### Remarks

The return value can be formatted using .NET functions.

## QuickScript .NET Variables

QuickScript .NET variables must be declared before they can be used in QuickScript .NET scripts. Variables can be used on both the left and right side of statements and expressions.

Local variables or attributes can be used together in the same script. Variables declared within the script body lose their value after the script is executed. Those declared in the script body cannot be accessed by other scripts.

Variables declared in the **Declarations** area maintain their values throughout the lifetime of the object that the script is associated with.

Each variable must be declared in the script by a separate DIM statement followed by a semicolon. Enter DIM statements in the **Declarations** area of the **Script** tab page. The DIM statement syntax is as follows:

```
DIM <variable_name> [ ( <upper_bound>
[, <upper_bound >[, < upper_bound >]] ) ]
[ AS <data_type> ];
```

where:

#### DIM

#### Required keyword.

<variable\_name>

Name that begins with a letter (A-Z or a-z) and whose remaining characters can be any combination of letters (A-Z or a-z), digits (0-9) and underscores (\_). The variable name is limited to 255 Unicode characters.

<upper_bound>	Reference to the upper bound (a number between 1 and 2,147,483,647, inclusive) of an array dimension. Three dimensions are supported in a <code>DIM</code> statement, each being nested in the syntax structure. After the upper bound is specified, it is fixed after the declaration. A statement similar to Visual Basic's <code>ReDim</code> is not supported.
The lower bound of each array dimension is always 1.	
AS	Optional keyword for declaring the variable's datatype.
<data_type>	Any one of the following 11 datatypes: Boolean, Discrete, Integer, ElapsedTime, Float, Real, Double, String, Message, Time or Object.  Data_type can also be a .Net data_type like System.Xml.XmlDocument or a type defined in an imported script library  If you omit the <code>AS</code> clause from the <code>DIM</code> statement, the variable, by default, is declared as an Integer datatype. For example:
<pre>DIM LocVar1;</pre> <p>is equivalent to:</p> <pre>DIM LocVar1 AS Integer;</pre>	

In contrast to attribute names, variable names must not contain dots. Variable names and the data type identifiers are not case sensitive. If there is a naming conflict between a declared variable and another named entity in the script (for example, attribute name, alias or name of an object leveraged by the script), the variable name takes precedence over the other named entities. If the variable name is the same as an alias name, a warning message appears when the script is validated to indicate that the alias is ignored.

The syntax for specifying the entire array is "[ ]" for both local array variables and for attribute references. For example, to assign an attribute array to a local array, the syntax is:

```
locarr[] = tag.attr[];
```

`DIM` statements can be located anywhere in the script body, but they must precede the first referencing script statement or expression. If a local variable is referenced before the `DIM` statement, script validation done when you save the object containing the script prompts you to define it.

**Caution:** The validation mentioned above occurs only when you save the object containing the script. This is not the script syntax validation done when you click the **Validate Script** button.

Do not cascade `DIM` statements. For example, the following examples are invalid:

```
DIM LocVar1 AS Integer, LocVar2 AS Real;
DIM LocVar3, LocVar4, LocVar5, AS Message;
```

To declare multiple variables, you must enter separate `DIM` statements for each variable.

When used on the right side of an equation, declared local variables always cause expressions on the left side to have Good quality. For example :

```
dim x as integer;
dim y as integer;
x = 5;
y = 5;
me.attr = 5;
me.attr = x;
```



```
me.attr = x+y;
```

In each case of `me.attr`, quality is Good.

When you use a variable in an expression to the right of the operator, its Quality is treated as Good for the purpose of data quality propagation.

You can use null to indicate that there is no object currently assigned to a variable. Using null has the same meaning as the keyword "null" in C# or "nothing" in Visual Basic. Assigning null to a variable makes the variable eligible for garbage collection. You may not use a variable whose value is null. If you do, the script terminates and an error message appears in the logger. You may, however, test a variable for null. For example:

```
IF myvar == null THEN ...
```

It is not possible to pass attributes as parameters for system objects. To work around this issue, use a local variable as an intermediary or explicitly convert the attribute to a string using an appropriate function call when calling the system object.

## Numbers and Strings

Allowed format for integer constants in decimal format is as follows:

```
IntegerConst = 0 or [sign] <non-zero_digit> <digit>*;
```

where:

```
sign ::= + | -
non-zero_digit ::= 1-9
digit ::= 0-9
```

For example, an integer constant is a zero or consists of an optional sign followed by one or more digits. Leading zeros are not allowed. Integer constants outside the range -2147483648 to 2147483647 cause an overflow error.

Prepending either 0x or 0X causes a literal integer constant to be interpreted as hexadecimal notation. The +/- sign is supported.

The acceptable float for integers in hexadecimal is as follows:

```
IntegerHexConst = [<sign>] <0><x (or X)> <hexdigit>*
```

where:

```
sign ::= + or -
hexdigit ::= 0-9, A-F, a-f (only eight hexdigits [32-bits] are allowed)
```

Allowed format for floats is as follows:

```
FloatConst ::= [<sign>] <digit>* .<digit>+ [<exponent>;]
```

or

```
[<sign>] <digit>+ [.<digit>* [<exponent>]];
```

where:

```
sign ::= + or -
digit ::= 0-9 (can be one or more decimal digits)
exponent = e (or E) followed by a sign and then digit(s)
```

Float constants are applicable as values for variables of type float, real, or double. For example, float constants do not take the number of bytes into account. Script validation detects an overflow when a float, real, or double variable has been assigned a float constant that exceeds the maximum value.

If no digits appear before the period (.), at least one must appear after it. If neither an exponent part nor the period appears, a period is assumed to follow the last digit in the string.

If an attribute reference exists that has a format similar to a float constant with an exponent (such as "5E3"), then use the Attribute qualifier, as follows:

```
Attribute("5E3")
```

Strings must be surrounded by double quotation marks. They are referred to as quoted strings. The double-double quote indicates a single double-quote in the string. For example, the string:

```
Joe said, "Look at that."
```

can be represented in QuickScript .NET as:

```
"Joe said, ""Look at that."""
```

## QuickScript .NET Control Structures

QuickScript .NET provides five primary control structures in the scripting environment:

- *IF ... THEN ... ELSEIF ... ELSE ... ENDIF* on page 394
- *FOR ... TO ... STEP ... NEXT Loop* on page 396
- *FOR EACH ... IN ... NEXT* on page 397
- *TRY ... CATCH* on page 397
- *WHILE Loop* on page 398

### IF ... THEN ... ELSEIF ... ELSE ... ENDIF

IF-THEN-ELSE-ENDIF conditionally executes various instructions based on the state of an expression. The syntax is as follows:

```
IF <Boolean_expression> THEN
    [statements];
[ { ELSEIF
    [statements] } ];
[ ELSE
    [statements] ];
ENDIF;
```

Where Boolean\_expression is an expression that can be evaluated as a Boolean.

Depending on the data type returned by the expression, the expression is evaluated to constitute a True or False state according to the following table:

Data Type	Mapping
Boolean, Discrete	Directly used (no mapping needed).
Integer	Value = 0 evaluated as False. Value != 0 evaluated as True.
Float, Real	Value = 0 evaluated as False. Value != 0 evaluated as True.
Double	Value = 0 evaluated as False. Value != 0 evaluated as True.
String, Message	Cannot be mapped. Using an expression that results in a string type as the Boolean_expression results in a script validation error.
Time	Cannot be mapped. Using an expression that results in a time type as the Boolean_expression results in a script validation error.
ElapsedTime	Cannot be mapped. Using an expression that results in an elapsed time type as the Boolean_expression results in a script validation error.

Data Type	Mapping
Object	Using an expression that results in an object type. Validates, but at run time, the object is converted to a Boolean. If the type cannot be converted to a Boolean, a run-time exception is raised.

The first block of statements is executed if Boolean\_expression evaluates to True. Optionally, a second block of statements can be defined after the keyword ELSE. This block is executed if the Boolean\_expression evaluates to False.

To help decide between multiple alternatives, an optional ELSEIF clause can be used as often as needed. The ELSEIF clause mimics switch statements offered by other programming languages. For example:

```
IF value == 0 Then
    Message = "Value is zero";
ELSEIF value > 0 Then
    Message = "Value is positive";
ELSEIF value < 0 Then
    Message = "Value is negative";
ELSE
    {Default. Should never occur in this example};
ENDIF;
```

The following approach nests a second IF compound statement within a previous one and requires an additional ENDIF:

```
IF (X1 == 1) THEN
    X1 = 5;
{ ELSEIF <X1 == 2> THEN
    X1 = 10;
ELSEIF X1 == 3 THEN
    X1 = 20 ;
ELSEIF X1 == 4 THEN
    X1 = 30 };
    IF X1 == 99 THEN
        X1 = 0;
    ENDIF;
ENDIF;
```

See Sample Scripts for more ideas about using this type of control structure.

## IF ... THEN ... ELSEIF ... ELSE ... ENDIF and Attribute Quality

When an attribute value is copied to another attribute of the same type, the attribute's quality is also copied. This can be especially relevant when working with I/O attributes. For example, the following two statements copy both value and quality:

```
me.Attr2 = me.Attr1;
me.Attr2.value = me.Attr1.value;
```

If only the value needs to be copied and the attribute has the quality BAD, you can use a temporary variable to hold the value. For example:

```
Dim temp as Integer;
temp = me.Attr1;
me.Attr2 = temp;
```

If there is a comparison such as Attr1 <> Attr2 and one of the attributes has the quality BAD, then the statements within the IF control block are not executed. For example, assuming Attr1 has the quality BAD:

```
if me.Attr1<> me.Attr2 then
    me.Attr2 = me.Attr1;
endif;
```

In this script, the statement `me.Attr2 = me.Attr1` is not executed because `Attr1` has the quality `BAD` and comparing a `BAD` quality value with a good quality value is not defined/not possible.

The recommended approach is to first verify the quality of `Attr1`, as shown in the following example:

```
if(IsBad(me.Attr1)) then
    LogMessage("Attr1 quality is bad, its value is not copied to Attr2");
else
    if me.Attr1<> me.Attr2 then
        me.AttrA2 = me.Attr1;
    endif;
endif;
```

An alternative method of verifying quality is to use the "==" operator:

```
if Me.Attr1 == TRUE then
```

Or, you can add the "value" property to the simplified IF THEN statement:

```
if Me.Attr1.value then
```

Using any of the above methods to verify data quality will ensure that your scripts execute correctly.

## FOR ... TO ... STEP ... NEXT Loop

FOR-NEXT performs a function (or set of functions) within a script several times during a single execution of a script. The general format of the FOR-NEXT loop is as follows:

```
FOR <analog_var> = <start_expression> TO <end_expression> [STEP
<change_expression>];
    [statements];
    [EXIT FOR;];
    [statements];
NEXT;
```

Where:

- `analog_var` is a variable of type Integer, Float, Real, or Double.
- `start_expression` is a valid expression to initialize `analog_var` to a value for execution of the loop.
- `end_expression` is a valid expression. If `analog_var` is greater than `end_expression`, execution of the script jumps to the statement immediately following the `NEXT` statement.

This holds true if loop is incrementing up, otherwise, if loop is decrementing, loop termination occurs if `analog_var` is less than `end_expression`.

- `change_expression` is an expression that defines the increment or decrement value of `analog_var` after execution of the `NEXT` statement. The `change_expression` can be either positive or negative.
  - If `change_expression` is positive, `start_expression` must be less than or equal to `end_expression` or the statements in the loop do not execute.
  - If `change_expression` is negative, `start_expression` must be greater than or equal to `end_expression` for the body of the loop to be executed.
- If `STEP` is not set, then `change_expression` defaults to 1 for increasing increments, and defaults to -1 for decreasing increments.

Exit the loop from within the body of the loop with the `EXIT FOR` statement.

The FOR loop is executed as follows:

1. `analog_var` is set equal to `start_expression`.

2. If `change_expression` is positive, the system tests to see if `analog_var` is greater than `end_expression`. If so, the loop exits. If `change_expression` is negative, the system tests to see if `analog_var` is less than `end_expression`. If so, program execution exits the loop.
3. The statements in the body of the loop are executed. The loop can potentially be exited via the EXIT FOR statement.
4. `analog_var` is incremented by 1,-1, or by `change_expression` if it is specified.
5. Steps 2 through 4 are repeated.

---

**Note:** FOR-NEXT loops can be nested. The number of levels of nesting possible depends on memory and resource availability.

---

## FOR EACH ... IN ... NEXT

FOR EACH loops can be used only with collections exposed by OLE Automation servers. A FOR-EACH loop performs a function (or set of functions) within a script several times during a single execution of a script. The general format of the FOR-EACH loop is as follows:

```
FOR EACH <object_variable> IN <collection_object >
  [statements];
  [EXIT FOR;];
  [statements];
NEXT;
```

Where:

- `object_variable` is a dimmed variable.
- `collection_object` is a variable holding a collection object.

As in the case of the FOR ... TO loop, it is possible to exit the execution of the loop through the statement EXIT FOR from within the loop.

## TRY ... CATCH

TRY ... CATCH provides a way to handle some or all possible errors that may occur in a given block of code, while still running rather than terminating the program. The TRY part of the code is known as the try block. Deal with any exceptions in the CATCH part of the code, known as the catch block.

The general format for TRY ... CATCH is as follows:

```
TRY
  [try statements] 'guarded section
CATCH
  [catch statements]
ENDTRY
```

Where:

*tryStatements*

Statement(s) where an error can occur. Can be a compound statement. The tryStatement is a guarded section.

*catchStatements*

Statement(s) to handle errors occurring in the associated Try block. Can be a compound statement.

---

**Note:** Statements inside the Catch block may reference the reserved ERROR variable, which is a .NET System.Exception thrown from the Try block. The statements in the Catch block run only if an exception is thrown from the Try block.

---

TRY ... CATCH is executed as follows:

1. Run-time error handling starts with TRY. Put code that might result in an error in the try block.
2. If no run-time error occurs, the script will run as usual. Catch block statements will be ignored.
3. If a run-time error occurs, the rest of the try block does not execute.
4. When a run-time error occurs, the program immediately jumps to the CATCH statement and executes the catch block.

The simplest kind of exception handling is to stop the program, write out the exception message, and continue the program.

The error variable is not a string, but a .NET object of System.Exception. This means you can determine the type of exception, even with a simple CATCH statement. Call the GetType() method to determine the exception type, and then perform the operation you want, similar to executing multiple catch blocks.

**Example:**

```
dim command = new System.Data.SqlClient.SqlCommand;
dim reader as System.Data.SqlClient.SqlDataReader;
command.Connection = new System.Data.SqlClient.SqlConnection;
try
    command.Connection.ConnectionString = "Integrated Security=SSPI";
    command.CommandText="select * from sys.databases";
    command.Connection.Open();
    reader = command.ExecuteReader();

    while reader.Read()
        me.name = reader.GetString(0);
        LogMessage(me.name);
    endwhile;
catch
    LogMessage(error);
endtry;
if reader <> null and not reader.IsClosed then
    reader.Close();
endif;
if command.Connection.State == System.Data.ConnectionState.Open then
    command.Connection.Close();
endif;
```

## WHILE Loop

WHILE loop performs a function or set of functions within a script several times during a single execution of a script while a condition is true. The general format of the WHILE loop is as follows:

```
WHILE <Boolean_expression>
    [statements]
    [EXIT WHILE;]
    [statements]
ENDWHILE;
```

Where: Boolean\_expression is an expression that can be evaluated as a Boolean as defined in the description of IF... THEN statements.

It is possible to exit the loop from the body of the loop through the EXIT WHILE statement.

The WHILE loop is executed as follows:

1. The script evaluates whether the Boolean\_expression is true or not. If not, program execution exits the loop and continues after the ENDWHILE statement.
2. The statements in the body of the loop are executed. The loop can be exited through the EXIT WHILE statement.
3. Steps 1 through 2 are repeated.

---

**Note:** WHILE loops can be nested. The number of levels of nesting possible depends on memory and resource availability.

---

## QuickScript .NET Operators

The following QuickScript .NET operators require a single operand:

Operator	Short Description
~	Complement
-	Negation
NOT	Logical NOT

The following QuickScript .NET operators require two operands:

Operator	Short Description
+	Addition and concatenation
-	Subtraction
&	Bitwise AND
*	Multiplication
**	Power
/	Division
^	Exclusive OR
	Inclusive OR
<	Less than
<=	Less than or equal to
<>	Not equal to
=	Assignment
==	Equivalency (is equivalent to); not supported for entire array compares. Arrays must be compared one element at a time using ==.
>	Greater than

Operator	Short Description
>=	Greater than or equal to
AND	Logical AND
MOD	Modulo
OR	Logical OR
SHL	Left shift
SHR	Right shift

The following table shows the precedence of QuickScript .NET operators:

Precedence	Operator
1 (highest)	( )
2	- (negation), NOT, ~
3	**
4	*, /, MOD
5	+, - (subtraction)
6	SHL, SHR
7	<, >, <=, >=
8	==, <>
9	&
10	^
11	
12	=
13	AND
14 (lowest)	OR

The arguments of the listed operators can be numbers or attribute values. Putting parentheses around an argument is optional. Operator names are not case-sensitive.

## Parentheses ( )

Parentheses specify the correct order of evaluation for the operator(s). They can also make a complex expression easier to read. Operator(s) in parentheses are evaluated first, preempting the other rules of precedence that apply in the absence of parentheses. If the precedence is in question or needs to be overridden, use parentheses.

In the example below, parentheses add B and C together before multiplying by D:

```
( B + C ) * D;
```



## Negation ( - )

Negation is an operator that acts on a single component. It converts a positive integer or real number into a negative number.

## Complement ( ~ )

This operator yields the one's complement of a 32-bit integer. It converts each zero-bit to a one-bit and each one-bit to a zero-bit. The one's complement operator is an operator that acts on a single component, and it accepts an integer operand.

## Power ( \*\* )

The Power operator returns the result of a number (the base) raised to the power of a second number (the power). The base and the power can be any real or integer numbers, subject to the following restrictions:

- A zero base and a negative power are invalid.  
Example: "0 \*\* - 2" and "0 \*\* -2.5"
- A negative base and a fractional power are invalid.  
Example: "-2 \*\* 2.5" and "-2 \*\* -2.5"
- Invalid operands yield a zero result.

The result of the operation should not be so large or so small that it cannot be represented as a real number. Example:

```
1 ** 1 = 1.0
3 ** 2 = 9.0
10 ** 5 = 100,000.0
```

## Multiplication ( \* ), Division ( / ), Addition ( + ), Subtraction ( - )

These binary operators perform basic mathematical operations. The plus (+) can also concatenate String datatypes.

For example, in the data change script below, each time the value of "Number" changes, "Setpoint" changes as well:

```
Number=1;
Setpoint.Name = "Setpoint" + Text(Number, "#" );
```

Where: The result is "Setpoint1."

## Modulo (MOD)

MOD is a binary operator that divides an integer quantity to its left by an integer quantity to its right. The remainder of the quotient is the result of the MOD operation. Example:

```
97 MOD 8 yields 1
63 MOD 5 yields 3
```

## Shift Left (SHL), Shift Right (SHR)

SHL and SHR are binary operators that use only integer operands. The binary content of the 32-bit word referenced by the quantity to the left of the operator is shifted (right or left) by the number of bit positions specified in the quantity to the right of the operator.

Bits shifted out of the word are lost. Bit positions vacated by the shift are zero-filled. The shift is an unsigned shift.

## Bitwise AND ( & )

A bitwise binary operator compares 32-bit integer words with each other, bit for bit. Typically, this operator masks a set of bits. The operation in this example "masks out" (sets to zero) the upper 24 bits of the 32-bit word. For example:

```
result = name & 0xff;
```

## Exclusive OR (^) and Inclusive OR (|)

The ORs are bitwise logical operators compare 32-bit integer words to each other, bit for bit. The Exclusive OR compare the status of bits in corresponding locations. If the corresponding bits are the same, a zero is the result. If the corresponding bits differ, a one is the result. Example:

```
0 ^ 0 yields 0
0 ^ 1 yields 1
1 ^ 0 yields 1
1 ^ 1 yields 0
```

The Inclusive OR examines the corresponding bits for a one condition. If either bit is a one, the result is a one. Only when both corresponding bits are zeros is the result a zero. For example:

```
0 | 0 yields 0
0 | 1 yields 1
1 | 0 yields 1
1 | 1 yields 1
```

## Assignment (=)

Assignment is a binary operator which accepts integer, real, or any type of operand. Each statement can contain only one assignment operator. Only one name can be on the left side of the assignment operator.

Read the equal sign (=) of the assignment operator as "is assigned to" or "is set to."

---

**Note:** Do not confuse the equal sign with the equivalency sign (==) used in comparisons.

---

## Comparisons (<, >, <=, >=, ==, <>)

Comparisons in IF-THEN-ELSE statements execute various instructions based on the state of an expression.

## AND, OR, and NOT

These operators work only on discrete attributes. If these operators are used on integers or real numbers, they are converted as follows:

- Real to Discrete: If real is 0.0, discrete is 0, otherwise discrete is 1.
- Integer to Discrete: If integer is 0, discrete is 0, otherwise discrete is 1.

If the statement is: "Disc1 = Real1 AND Real2;" and Real1 is 23.7 and Real2 is 0.0, Disc1 has 0 assigned to it, since Real1 is converted to 1 and Real2 is converted to 0.

When assigning the floating-point result of a mathematical operation to an integer, the value is rounded to the nearest integer instead of truncating it. This means that an operation like `IntAttr = 32/60` results in `IntAttr` having a value of 1, not 0. If truncation is needed, use the `Trunc()` function.