



Wonderware
Creating and
Managing ArchedrA
Graphics User's Guide

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 Schneider Electric Software, LLC. 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, Schneider Electric Software, LLC 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 Schneider Electric Software, LLC. 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.

© 2015 Schneider Electric Software, LLC. All rights reserved.

Schneider Electric Software, LLC
26561 Rancho Parkway South
Lake Forest, CA 92630 U.S.A.
(949) 727-3200

<http://software.schneider-electric.com/>

ArchestrA, Avantis, DYNsIM, EYESIM, Foxboro, Foxboro Evo, I/A Series, InBatch, InduSoft, IntelaTrac, InTouch, PIPEPHASE, PRO/II, PROVISION, ROMeo, Schneider Electric, SIM4ME, SimCentral, SimSci, Skelta, SmartGlance, Spiral Software, VISUAL FLARE, WindowMaker, WindowViewer, and Wonderware are trademarks of Schneider Electric SE, its subsidiaries, and affiliated companies. An extensive listing of Schneider Electric Software, LLC trademarks can be found at: <http://software.schneider-electric.com/legal/trademarks/>. All other brands may be trademarks of their respective owners.

Contents

| | | |
|------------------|--|-----------|
| | Welcome | 17 |
| | Documentation Conventions | 17 |
| | Technical Support | 18 |
| Chapter 1 | About Creating and Managing ArchestrA Symbols | 19 |
| | Managing ArchestrA Symbols | 20 |
| | Managing Symbols in the Graphic Toolbox | 20 |
| | Managing Symbols in Automation Objects | 21 |
| | Re-using ArchestrA Symbols | 21 |
| | ArchestrA Symbols and Instantiation | 22 |
| | ArchestrA Symbol Creation: The ArchestrA Symbol Editor | 23 |
| | The ArchestrA Symbol Editor | 23 |
| | Elements | 25 |
| | Properties | 30 |
| | Animations | 34 |
| | Embedded Symbols | 39 |
| | Appearance of Embedded Symbols | 39 |
| | Changing Embedded Symbols | 40 |
| | Embedding and Instantiation | 41 |
| | Symbol Change Propagation | 42 |
| | Size Propagation and Anchor Points | 43 |

| | | |
|------------------|--|-----------|
| | Estimating Graphic Performance | 44 |
| | Estimating Symbol Performance | 45 |
| | Understanding GPI Rating Calculations | 46 |
| | Examining a Symbol with a 4.5 GPI Rating | 51 |
| | Saving a Symbol that May Impact Run-time Performance | 52 |
| | Showing Quality and Status | 53 |
| | Showing Quality and Status with the Status Element | 53 |
| | Showing Quality and Status by Overriding | 53 |
| Chapter 2 | Comparing WindowMaker and ArchestrA Symbol Editor | 55 |
| | Differences between WindowMaker and the ArchestrA Symbol Editor | 55 |
| | Elements | 55 |
| | Appearance | 56 |
| | Enhanced Functionality | 56 |
| | Procedures for Common WindowMaker Tasks and Techniques | 58 |
| | Using Graphics | 58 |
| | Using Animations | 60 |
| | Using Scripts | 65 |
| Chapter 3 | Managing Symbols | 67 |
| | About Symbols | 67 |
| | Creating a New Symbol | 68 |
| | Creating Symbols in the Graphic Toolbox | 68 |
| | Creating Symbols in AutomationObject Templates | 69 |
| | Creating Symbols in AutomationObject Instances | 69 |
| | Opening Symbols for Editing | 70 |
| | Organizing Symbols in the Graphic Toolbox | 71 |
| | Creating Graphic Toolsets in the Graphic Toolbox | 71 |
| | Moving Symbols between Graphic Toolsets | 72 |
| | Renaming Symbols | 72 |
| | Copying Symbols | 72 |
| | Renaming Graphic Toolsets | 73 |
| | Deleting Graphic Toolsets | 73 |
| | Moving Graphic Toolsets | 73 |
| | Customizing Graphic Toolsets | 74 |
| | Importing and Exporting Symbols as ArchestrA Object Files | 74 |
| | Importing Symbols | 74 |
| | Exporting Symbols | 75 |

| | |
|---|--|
| Programmatically Importing and Exporting | |
| ArchestrA Symbols | 76 |
| Implementing the GraphicAccess API | 77 |
| Importing ArchestrA Symbols from XML Files | 79 |
| Exporting ArchestrA Symbols to XML Files | 79 |
| Exporting and Importing Overridden Text Strings | 79 |
| Associating All Galaxy Graphics with an InTouchEventApp | 83 |
| Deleting a Symbol | 84 |
| Creating Multiple Configurations of a Symbol | 84 |
| Understanding Visual and Functional Symbol | |
| Configurations | 85 |
| Different Symbol Wizard Work Flows | 86 |
| Embedded Symbols | 86 |
| Appearance of Embedded Symbols | 86 |
| Changing Embedded Symbols | 87 |
| Configuring Security for Symbols | 87 |
| Writing to Attributes Configured for Secured or | |
| Verified Writes | 89 |
| Working with the SignedWrite() Function for | |
| Secured and Verified Writes | 89 |
| SignedWrite() Run-time Behavior | 90 |
| SignedWrite() Scripting Tips | 91 |
| Examples of Using the Attribute Parameter in the | |
| SignedWrite() Function | 92 |
| Secured and Verified Write Applied Examples | 94 |
| Viewing a Symbol in Read-Only Mode | 96 |
| | |
| Chapter 4 | Using the ArchestrA Symbol Editor 97 |
| Showing, Hiding, and Adjusting Panels | 98 |
| Panning and Zooming the Canvas | 98 |
| Panning | 98 |
| Zooming | 100 |
| Configuring Designer Preferences | 103 |
| Using the Symbol Wizard Editor | 105 |
| | |
| Chapter 5 | Working with Graphic Elements 109 |
| About Graphic Elements | 109 |
| Drawing and Dragging Elements | 110 |
| Drawing Rectangles, Rounded Rectangles, | |
| Ellipses, and Lines | 110 |
| Drawing Polylines, Polygons, Curves, and Closed Curves | 111 |
| Drawing 2-Point Arcs, 2-Point Pies and 2-Point Chords | 111 |

| | |
|--|-----|
| Drawing 3-Point Arcs, 3-Point Pies, and 3-Point Chords | 112 |
| Placing and Importing Images | 112 |
| Drawing Buttons | 112 |
| Placing Text | 113 |
| Drawing Text Boxes | 113 |
| Drawing Status Elements | 113 |
| Drawing Windows Controls | 114 |
| Dragging Elements | 114 |
| Editing Element Properties | 114 |
| Selecting Elements | 115 |
| Selecting Elements by Mouse Click | 116 |
| Selecting Elements by Lasso | 116 |
| Selecting All Elements | 117 |
| Selecting Elements Using the Elements List | 117 |
| Unselecting Elements | 118 |
| Inline Editing | 118 |
| Copying, Cutting, and Pasting Elements | 119 |
| Copying Elements | 120 |
| Cutting or Deleting Elements | 120 |
| Duplicating Elements | 121 |
| Moving Elements | 121 |
| Aligning Elements | 123 |
| Aligning Elements Horizontally | 123 |
| Aligning Elements Vertically | 124 |
| Aligning Elements by their Center Points | 124 |
| Aligning Elements by their Points of Origin | 125 |
| Adjusting the Spacing between Elements | 125 |
| Distributing Elements | 126 |
| Making Space between Elements Equal | 126 |
| Increasing Space between Elements | 127 |
| Decreasing Space between Elements | 127 |
| Removing All Space between Elements | 128 |
| Resizing Elements | 128 |
| Resizing a Single Element with the Mouse | 129 |
| Resizing Elements by Changing Size Properties | 129 |
| Resizing Elements Proportionally | 130 |
| Making Elements the Same Width, Height, or Size | 130 |
| Adjusting the z-Order of Elements | 131 |
| Rotating Elements | 132 |
| Rotating Elements with the Mouse | 132 |
| Rotating Elements by Changing the Angle Property | 133 |
| Rotating Elements by 90 Degrees | 133 |

| | |
|--|------------|
| Moving the Origin of an Element | 134 |
| Changing Points of Origin with the Mouse | 134 |
| Changing Points of Origin in the Properties Editor | 134 |
| Adding Connectors Between Graphic Elements | 135 |
| Drawing a Connector | 136 |
| Adding Connection Points | 138 |
| Using Custom Connection Points with Embedded and Grouped Symbols | 139 |
| Using Connection Points with Embedded Situational Awareness Library Symbols | 139 |
| Changing Connector Properties | 142 |
| Flipping Elements | 148 |
| Locking and Unlocking Elements | 148 |
| Making Changes Using Undo and Redo | 149 |
| Working with Groups of Elements | 150 |
| Creating a Group of Elements | 150 |
| Ungrouping | 151 |
| Adding Elements to Existing Groups | 151 |
| Removing Elements from Groups | 152 |
| Editing Components within a Group | 152 |
| Using Path Graphics | 153 |
| Creating a Path Graphic | 154 |
| Breaking the Path of a Path Graphic | 155 |
| Changing a Path Graphic | 155 |
| Adding Elements to an Existing Path Graphic | 160 |
| Removing Elements from a Path Graphic | 161 |
| | |
| Chapter 6 Editing Common Properties of Elements and Symbols | 163 |
| Editing the Name of an Element | 164 |
| Editing the Fill Properties of an Element | 164 |
| Setting Fill Style | 165 |
| Setting Unfilled Style | 166 |
| Setting Fill Orientation | 166 |
| Setting Fill Behavior | 167 |
| Setting Horizontal Fill Direction and Percentage | 167 |
| Setting Vertical Fill Direction and Percentage | 168 |
| Editing the Line Properties of an Element | 168 |
| Setting Start or End Points of a Line | 169 |
| Setting the Line Weight | 169 |
| Setting the Line Pattern | 169 |
| Setting the Line Style | 170 |

| | |
|---|------------|
| Setting the Text Properties of an Element | 171 |
| Setting the Displayed Text | 171 |
| Setting the Text Display Format | 171 |
| Setting the Text Font | 172 |
| Setting the Text Color | 172 |
| Setting the Text Alignment | 173 |
| Substituting Strings | 174 |
| Setting Style | 176 |
| Setting a Solid Color | 176 |
| Setting a Gradient | 180 |
| Setting a Pattern | 184 |
| Setting a Texture | 185 |
| Setting the Style to No Fill | 186 |
| Setting the Transparency of a Style | 186 |
| Setting the Transparency Level of an Element | 186 |
| Tweaking the Colors and Transparency of a Gradient | 187 |
| Loading Graphics with Deprecated Features | 188 |
| Enabling and Disabling Elements for Run-Time Interaction | 188 |
| Changing the Visibility of Elements | 189 |
| Editing the Tab Order of an Element | 189 |
| Using the Format Painter to Format Elements | 190 |
| Editing the General Properties of a Symbol | 191 |
| | |
| Chapter 7 Working with Element Styles | 193 |
| Understanding Element Styles | 193 |
| Galaxy Style Library | 194 |
| Visual Properties Defined by Element Styles | 194 |
| Element Styles in Animations | 195 |
| Property Style Order of Precedence | 195 |
| Updating Element Styles at Application Run Time | 195 |
| Managing Element Styles | 195 |
| Importing and Exporting Galaxy Style Libraries | 195 |
| Changing Visual Properties of an Element Style | 196 |
| Working with User-Defined Element Styles | 200 |
| Applying Element Styles to Elements | 201 |
| Using the Element Style List | 201 |
| Using the Properties Grid | 202 |
| Using Format Painter | 202 |
| Clearing an Element Style | 202 |
| Selecting an Element Style as a Default for a Canvas | 203 |
| Applying Element Styles to Groups of Elements | 203 |

| | |
|--|-----|
| Setting a Group's Run-time Behavior to TreatAsIcon | 203 |
| Understanding Element Style Behavior with a Group of Elements | 204 |
| Configuring an Animation Using Element Styles | 204 |
| Configuring a Boolean Animation Using Element Styles | 204 |
| Configuring a Truth Table Animation with Element Styles | 205 |
| | |
| Chapter 8 Setting Symbol and Element-Specific Properties | 209 |
| Setting the Radius of Rounded Rectangles | 210 |
| Setting Line End Shape and Size | 211 |
| Setting Auto Scaling and Word Wrapping for a Text Box | 212 |
| Using Images | 212 |
| Placing an Image on the Canvas | 213 |
| Setting the Image Display Mode | 213 |
| Setting the Image Alignment | 214 |
| Setting the Image Color Transparency | 214 |
| Editing the Image | 215 |
| Setting the Image Editing Application | 215 |
| Selecting a Different Image | 216 |
| Using Buttons | 216 |
| Automatically Scaling Text in Buttons | 216 |
| Wrapping Text in Buttons | 217 |
| Configuring Buttons with Images | 217 |
| Editing Control Points | 218 |
| Moving Control Points | 218 |
| Adding and Removing Control Points | 218 |
| Changing the Tension of Curves and Closed Curves | 219 |
| Changing Angles of Arcs, Pies and Chords | 219 |
| Utilizing Sweep Angle Run-Time Properties | 220 |
| Monitoring and Showing Quality and Status | 221 |
| Using Status Elements | 221 |
| Overriding Element Appearance Depending on Quality and Status of its Attributes | 224 |
| Setting Global Number Styles | 228 |
| Configuring Global Number Styles | 229 |
| Working with User-defined Global Number Styles | 230 |
| Setting Number Formats by Regional Locales | 231 |
| Design Time Considerations for Numeric Formatting | 231 |
| Run-Time Considerations for Formatting Numbers | 236 |
| Restrictions of Numeric Formatting by Regional Locale | 237 |

| | |
|--|------------|
| Using Windows Common Controls | 238 |
| Changing Background Color and Text Color of Windows Common Controls | 239 |
| Reading and Writing the Selected Value at Run Time | 239 |
| Configuring Radio Button Group Controls | 240 |
| Configuring Check Box Controls | 242 |
| Configuring Edit Box Controls | 243 |
| Configuring Combo Box Controls | 244 |
| Configuring Calendar Controls | 246 |
| Configuring DateTime Picker Controls | 249 |
| Configuring List Box Controls | 251 |
| | |
| Chapter 9 Using Custom Properties | 253 |
| About Custom Properties | 254 |
| Managing Custom Properties | 254 |
| Adding and Deleting Custom Properties | 254 |
| Configuring Custom Properties | 255 |
| Validating Custom Properties | 257 |
| Clearing the Configuration of Custom Properties | 257 |
| Renaming Custom Properties | 257 |
| Linking Custom Properties to External Sources | 258 |
| Overriding Custom Properties | 258 |
| Reverting to Original Custom Property Values | 259 |
| Examples of Using Custom Properties | 259 |
| Using Custom Properties to Show Historical Summary Data | 259 |
| Analog Statistical Summary Data | 260 |
| State Statistical Summary Data | 261 |
| Historical Summary Period | 262 |
| Showing Statistical Summary Data | 263 |
| Using Binding in Custom Properties | 265 |
| Changing the Expression or Reference of a Custom Property at Run Time | 267 |
| | |
| Chapter 10 Animating Graphic Elements | 269 |
| Adding an Animation to an Element | 270 |
| Reviewing which Animations are Assigned to an Element | 270 |
| Showing and Hiding the Animation List | 271 |
| Removing Animations from an Element | 271 |
| Enabling and Disabling Animations | 272 |
| Validating the Configuration of an Animation | 272 |

| | |
|---|-----|
| Clearing the Configuration from an Animation | 273 |
| Connecting Animations with Data Sources | 274 |
| Connecting Animations with Arcestra Attributes | 274 |
| Connecting Animations with Element Properties | 275 |
| Connecting Animations with Custom Properties | 276 |
| Connecting Animations with InTouch Tags | 277 |
| Setting the Input Mode | 280 |
| Managing Animations | 280 |
| Organizing the Animation List | 280 |
| Switching between Animations | 281 |
| Configuring Common Types of Animations | 281 |
| Configuring a Visibility Animation | 282 |
| Configuring a Fill Style Animation | 282 |
| Configuring a Line Style Animation | 285 |
| Configuring a Text Style Animation | 289 |
| Configuring a Blink Animation | 292 |
| Configuring an Alarm Border Animation | 293 |
| Configuring a Percent Fill Horizontal Animation | 301 |
| Configuring a Percent Fill Vertical Animation | 303 |
| Configuring a Horizontal Location Animation | 305 |
| Configuring a Vertical Location Animation | 306 |
| Configuring a Width Animation | 306 |
| Configuring a Height Animation | 307 |
| Configuring a Point Animation | 308 |
| Configuring an Orientation Animation | 309 |
| Configuring a Value Display Animation | 311 |
| Configuring a Tooltip Animation | 316 |
| Configuring a Disable Animation | 317 |
| Configuring a User Input Animation | 318 |
| Configuring a Horizontal Slider Animation | 325 |
| Configuring a Vertical Slider Animation | 326 |
| Configuring a Pushbutton Animation | 327 |
| Configuring an Action Script Animation | 330 |
| Configuring a Show Symbol Animation | 332 |
| Configuring a Hide Symbol Animation | 340 |
| Configuring Element-Specific Animations | 341 |
| Configuring Animation for a Status Element | 341 |
| Configuring a Radio Button Group Animation | 342 |
| Configuring a Check Box Animation | 345 |
| Configuring an Edit Box Animation | 346 |
| Configuring a Combo Box Animation | 347 |
| Configuring a Calendar Control Animation | 350 |

| | |
|--|------------|
| Configuring a DateTime Picker Animation | 351 |
| Configuring a List Box Animation | 352 |
| Configuring a Trend Pen | 355 |
| Submitting the Value Changes | 363 |
| Format Strings in Element-Specific Animations | 363 |
| Cutting, Copying and Pasting Animations | 366 |
| Substituting References in Elements | 367 |
| | |
| Chapter 11 Adding and Maintaining Symbol Scripts | 369 |
| About Symbol Scripts | 369 |
| Predefined and Named Scripts | 370 |
| Execution Order of Symbol Scripts | 370 |
| Security in Symbol Scripts | 371 |
| Signature Security for Acknowledging Alarms | 371 |
| SignedAlarmAck() Run-time Behavior | 371 |
| Symbol Script Time outs | 375 |
| Error Handling | 376 |
| Configuring the Predefined Scripts of a Symbol | 376 |
| Ensuring Proper OnShow Script Execution | 377 |
| Adding Named Scripts to a Symbol | 378 |
| Editing Symbol Scripts | 379 |
| Renaming Scripts in a Symbol | 380 |
| Removing Scripts from a Symbol | 380 |
| Substituting Attribute References in Scripts | 381 |
| Example of Changing Element Properties using Scripts | 381 |
| Using Methods in Scripting | 382 |
| Configuring Edit Box Methods | 382 |
| Configuring Combo Box and List Box Methods | 383 |
| | |
| Chapter 12 Using Client Controls | 387 |
| About Client Controls | 388 |
| Importing Client Controls | 389 |
| Importing Client Controls | 389 |
| Importing Previously Exported Client Controls | 390 |
| Organizing Client Controls | 391 |
| Embedding Client Controls | 391 |
| Example of Embedding the ActiveFactory TagPicker Client Control | 391 |
| Viewing and Changing the Properties of Client Controls | 392 |
| Example of Changing a Property of the ActiveFactory TagPicker Control | 393 |

| | |
|---|-----|
| Binding Client Control Properties to Attributes or Element References | 393 |
| Example of Data Binding in the ActiveFactory TagPicker Control | 394 |
| Configuring Client Control Event Scripts | 395 |
| Example of Configuring an Event Script for the ActiveFactory TagPicker Control | 396 |
| Animating Client Controls | 397 |
| Exporting Client Controls | 398 |
| Securing Client Controls | 398 |
| Including Dynamically Loaded Assemblies with the Client Control | 399 |
| Requirements for Both Inclusion Methods | 399 |
| Sample XML for a Dynamically Loaded Assembly List | 400 |
| XML Schema for the Dynamically Loaded Assembly List | 400 |
| Embedding the XML Manifest Resource in the Primary Assembly | 401 |
| Including the XML Manifest Resource in an External Configuration File | 401 |
| Preventing Dynamically Loaded Assembly Import Issues | 402 |
| Viewing Additional Client Control Information | 402 |
| Viewing the Client Control Assemblies | 403 |
| Viewing Class Name, Vendor, and Version of a Client Control | 403 |
| Viewing Objects and Symbols Referencing Client Controls ... | 403 |
| Chapter 13 Embedding Symbols within Symbols | 405 |
| Embedding Symbols | 406 |
| Renaming Source Symbols and Hosting AutomationObjects ... | 407 |
| Editing the Embedded Symbol | 408 |
| Overriding Custom Properties of the Source Symbol | 409 |
| Restoring an Embedded Symbol to the Original Size of its Source Symbol | 410 |
| Converting an Embedded Symbol to a Group | 410 |
| Detecting the Source Symbol of an Embedded Symbol | 411 |
| Editing the Source of an Embedded Symbol | 411 |
| Controlling Size Propagation of Embedded Symbols | 412 |
| Setting the Anchor Point of a Source Symbol | 412 |
| Showing or Hiding the Anchor Points of Embedded Symbols | 413 |
| Enabling or Disabling Dynamic Size Change of Embedded Symbols | 413 |

| | |
|--|-----|
| Selecting Alternate Symbols and Instances | 414 |
| Selecting Alternate Symbols | 414 |
| Selecting Alternate Instances | 415 |
| Detecting and Editing the Containing | |
| AutomationObject Instance | 415 |
| Creating a New Instance of the Containing | |
| AutomationObject | 416 |
| | |
| Chapter 14 Migrating InTouch SmartSymbols..... | 417 |
| Importing InTouch SmartSymbols into an | |
| ArchestrA Symbol | 417 |
| Restrictions for SmartSymbol Import | 419 |
| Importing InTouch Graphics | 419 |
| Importing Graphical Animation | 421 |
| Importing Action Scripts | 422 |
| Importing References | 423 |
| | |
| Chapter 15 Switching Languages for | |
| Graphic Elements..... | 425 |
| About Language Switching for ArchestrA Graphics | 425 |
| Graphic Elements that Support Translation | 426 |
| Animations that Support Translation | 427 |
| Selecting the Language for a Symbol | 428 |
| Removing a Language for a Symbol | 429 |
| Creating Elements When Multiple Languages | |
| are Defined for a Galaxy | 429 |
| Moving Symbols to Galaxies with Different | |
| Language Settings | 430 |
| How Fonts are Applied at Design Time | 430 |
| Language Switching for Embedded Symbols | 431 |
| String Substitutions and Language Switching | 432 |
| Translating String Custom Properties | 433 |
| Translating Custom Properties for a Base Symbol | 434 |
| Translating Custom Properties for an Embedded Symbol | 434 |
| Translation Support for Client Controls with | |
| Satellite Assemblies | 435 |
| Translation Support for ArchestrA Client Controls | 435 |
| Importing InTouch SmartSymbols that Have | |
| Translated Data | 436 |
| Support for Empty Strings | 436 |
| Language Switching Example | 437 |

| | |
|---|-----|
| Overriding Translated Strings for Arcestra Symbols in WindowMaker | 440 |
| Overriding Translated String Substitutions | 440 |
| Overriding Translated Custom Properties | 440 |
| Language Switching at Run Time | 440 |
| How Languages are Shown in WindowViewer | 441 |
| Precedence Rules for Showing the Language and Font | 442 |
| Default Language Fonts at Run Time | 442 |
| Switching Languages for Custom Properties at Run Time | 443 |
| Switching Languages and String Substitutions at Run Time | 444 |
| Language Settings for Popup Symbols | 445 |
| Language Settings and Data Types | 446 |
| | |
| Chapter 16 Working with the Show/Hide Graphics Script Functions | 447 |
| About the Show/Hide Graphic Functions | 447 |
| Configuring the Show/Hide Graphic Script Functions | 448 |
| Using the Display Graphic Browser and Display Automation Object Browser | 449 |
| Show/Hide Graphic Script Functions Guidelines | 450 |
| Using the Show/Hide Script Parameters and Properties | 450 |
| Run Time Behavior of the Show/Hide Graphic Functions | 457 |
| Behavior of ShowGraphic Windows with the Same Identity | 458 |
| Closing a Symbol | 458 |
| Show/Hide Graphic Script Tips and Examples | 459 |
| Using Predefined and Named Scripts | 459 |
| Working with Modal Windows | 461 |
| Using Hierarchical References and Containment Relationships | 463 |
| Scripting the Owing Object | 464 |
| Assigning Custom Property Values of a Symbol | 469 |
| Scripting Multiple Symbols | 470 |
| | |
| Chapter 17 Working with Symbol Wizards | 475 |
| Understanding the Symbol Wizard Editor | 476 |
| Understanding Choice Groups and Choices | 476 |
| Understanding Symbol Wizard Layers | 477 |
| Defining Symbol Configuration Rules | 478 |
| Designing a Symbol Wizard | 480 |
| Creating Symbol Choice Groups, Choices, and Options | 480 |

| | |
|---|-----|
| Assigning Symbol Configuration Rules | 481 |
| Updating Symbol Layers | 482 |
| Associating Configuration Elements to Symbol Layers | 484 |
| Verify Symbol Configurations | 489 |
| Using Symbol Wizards in an Application | 489 |
| Embedding Symbol Wizards | 490 |
| Configuring Symbol Wizards in WindowMaker | 491 |
| Symbol Wizard Tips and Examples | 494 |
| Creating Visual Configurations of an ArcestraA Symbol | 494 |
| | |
| Appendix A List of Element Properties | 505 |
| Alphabetical List of Properties | 505 |
| List by Functional Area | 529 |
| Graphic Category Properties | 530 |
| Appearance Category Properties | 530 |
| Fill Style Group Properties | 542 |
| Line Style Group Properties | 545 |
| Text Style Group Properties | 547 |
| Runtime Behavior Group Properties | 549 |
| Custom Properties Group Properties | 553 |
| Order of Precedence for Property Styles | 554 |
| | |
| Appendix B Windows Common Control List Methods | 555 |
| Overview of Windows Common Control List Methods | 555 |
| | |
| Appendix C Situational Awareness Library Symbols | 559 |
| Common Graphic Elements of Situational Awareness Library Symbols | 559 |
| | |
| Index | 565 |

Welcome

You can use the ArcestrA[®] Integrated Development Environment (IDE) to create symbols with the ArcestrA Symbol Editor that appear in managed InTouch[®] applications. This document explains how to use the Symbol Editor to create and manage ArcestrA graphics.

You can view this document online [Creating and Managing ArcestrA Graphics User's Guide](#) or you can print it, in part or whole, by using the print feature in Adobe Reader.

This documentation assumes you know how to use Microsoft Windows, including navigating menus, moving from application to application, and moving objects on the screen. If you need help with these tasks, see the Microsoft documentation.

Documentation Conventions

This documentation uses the following conventions:

| Convention | Used for |
|-------------------|--|
| Initial Capitals | Paths and file names. |
| Bold | Menus, commands, dialog box names, and dialog box options. |
| Monospace | Code samples and display text. |

Technical Support

Wonderware Technical Support offers a variety of support options to answer any questions on Wonderware products and their implementation.

Before you contact Technical Support, refer to the relevant section(s) in this documentation for a possible solution to the problem. If you need to contact technical support for help, have the following information ready:

- The type and version of the operating system you are using.
- Details of how to recreate the problem.
- The exact wording of the error messages you saw.
- Any relevant output listing from the Log Viewer or any other diagnostic applications.
- Details of what you did to try to solve the problem(s) and your results.
- If known, the Wonderware Technical Support case number assigned to your problem, if this is an ongoing problem.

Chapter 1

About Creating and Managing ArchestrA Symbols

ArchestrA symbols are graphics you can create to visualize data in an InTouch HMI system.

You use the ArchestrA Symbol Editor to create ArchestrA symbols from basic elements, such as rectangles, lines, and text elements.

After you create an ArchestrA symbol, you can embed it into another symbol or an InTouch window and use it at run time.

You can embed an ArchestrA symbol in a template or instance of an ArchestrA Object, providing a way to visualize object-specific information quickly and easily. Embedding a symbol in a template means that you can update one symbol and cascade the changes throughout your application.

The ArchestrA Symbol Editor is a powerful addition to the standard InTouch editor, called WindowMaker.

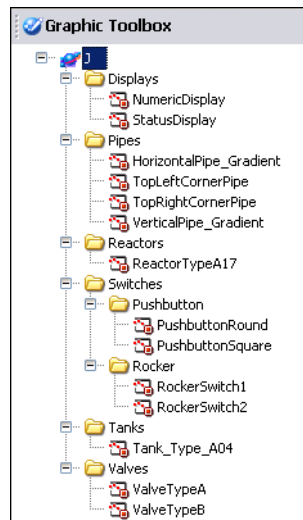
Managing ArcestrA Symbols

Depending on your development requirements, you can select where and how to store ArcestrA symbols.

- Store symbols in the Graphic Toolbox if you want to define them as a standard that you can re-use, such as a generic valve symbol. You can store ArcestrA symbols here if you only want to use them in the InTouch HMI.
- Store symbols as AutomationObject templates if you want to use the symbols in multiple instances at run time. For example, you can create a valve symbol contained in an AutomationObject template that represents the functionality of a valve type on your plant floor.
- Store symbols as AutomationObject instances if you want to use the symbols in only one specific object instance. For example, an AutomationObject instance that can be assigned a very specific piece of machinery as a symbol.

Managing Symbols in the Graphic Toolbox

The Graphic Toolbox enables you to organize your symbols in special folders called toolsets. You can create a hierarchy of toolsets. You can also move symbols between toolsets.



Note: Symbol names must be unique within the Graphic Toolbox.

Managing Symbols in Automation Objects

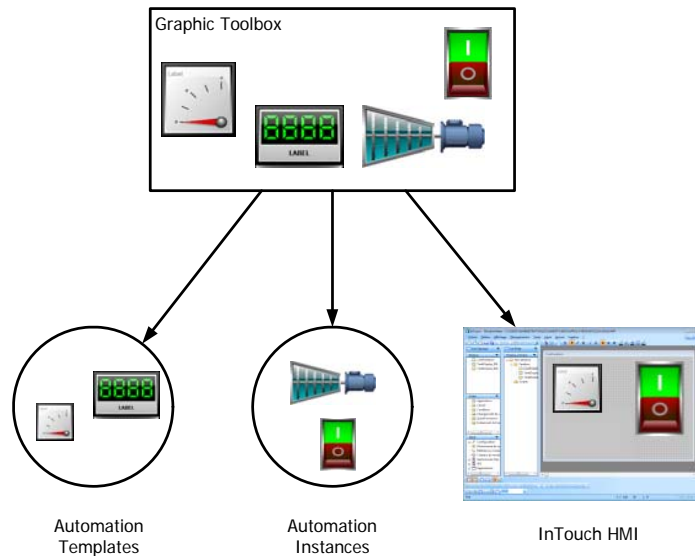
You can create ArchestrA symbols in AutomationObjects. Each AutomationObject has a **Graphics** tab to create, edit, rename, and delete ArchestrA symbols that belong to an AutomationObject. These symbols appear in the **Local Graphics** list.

When you derive an AutomationObject from a parent AutomationObject that contains symbols, all the symbols are inherited. Inherited symbols appear in the **Inherited Graphics** list.

Note: You can only open inherited symbols in the ArchestrA Symbol Editor in read-only mode.

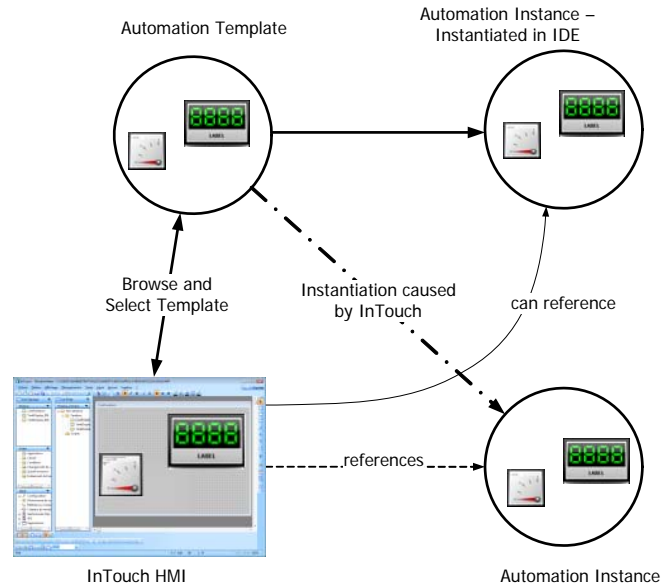
Re-using ArchestrA Symbols

You can re-use ArchestrA symbols that you create with AutomationObject templates, AutomationObject instances, or in InTouch windows. This is called embedding.



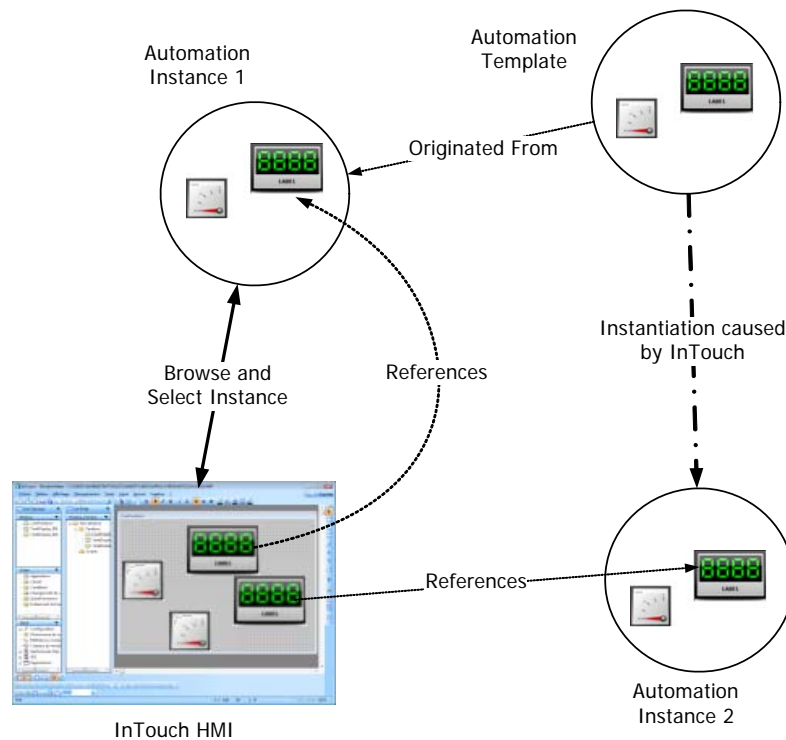
When you derive an AutomationObject template, its ArchestrA symbols are inherited by the new instance. This can be caused by:

- Deriving an instance of the template in the IDE. When you derive an instance of an AutomationObject template that contains symbols, the created instance contains inherited symbols.
- Embedding a new ArchestrA symbol in WindowMaker. A new AutomationObject instance is derived to which the symbol in InTouch WindowMaker then points.



ArcestrA Symbols and Instantiation

When you embed an ArcestrA symbol into an InTouch window and the symbol is contained in an AutomationObject template, you can easily create a new instance of the AutomationObject. The embedded ArcestrA symbol automatically references the new object.



ArchestrA Symbol Creation: The ArchestrA Symbol Editor

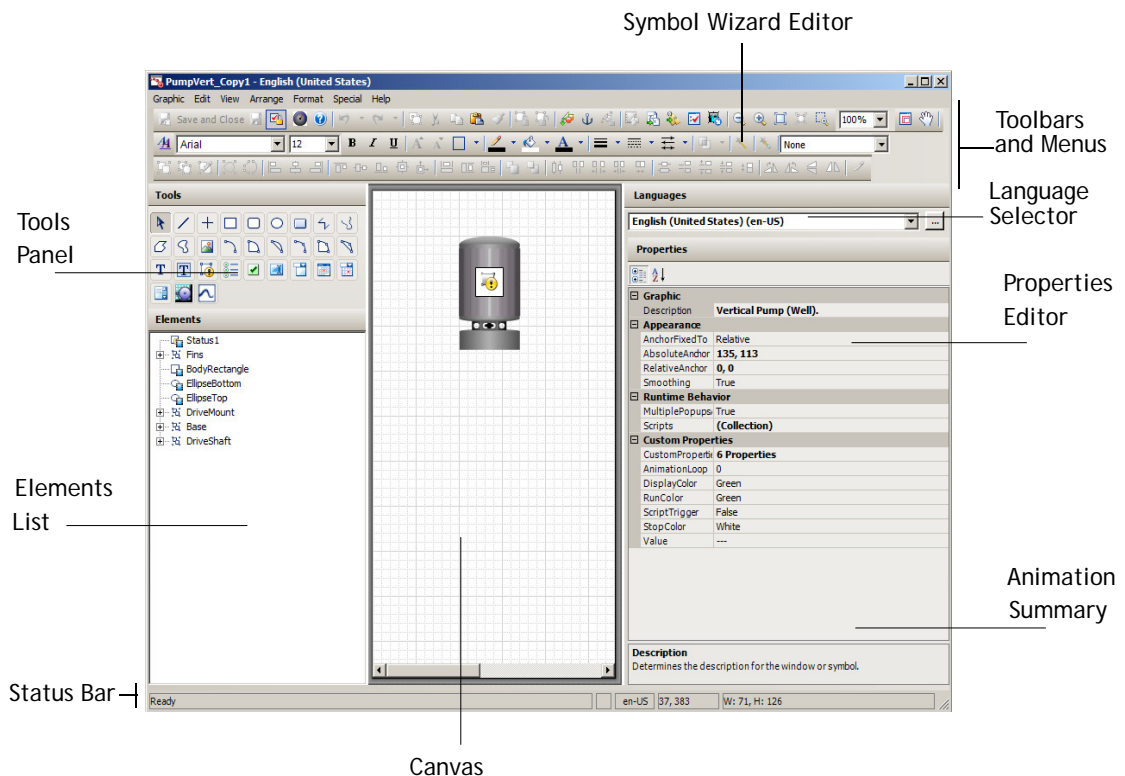
You use the ArchestrA Symbol Editor to create an ArchestrA symbol. First, you select a basic graphical object, called an element, from a tools panel and place it on the drawing area, called the canvas. Typical elements are lines, rectangles, ellipses, curves, and so on.

You can then change the appearance of your drawn elements by accessing their properties directly or by graphically manipulating them.

Finally, you can configure animations for the elements or the symbol.

The ArchestrA Symbol Editor

After you open the ArchestrA Symbol Editor, you see the various tools and palettes to create and customize symbols.



The ArchestrA Symbol Editor includes the following areas:

- **Tools Panel.** This is a collection of graphic elements to create your symbol.
- **Canvas.** This is the area in which you place and edit elements to create a symbol.

- **Elements List.** This list shows the named elements on the canvas in a hierarchical view.
- **Language Selector.** This list shows the configured languages for the symbol. For more information, see Chapter 15, "Switching Languages for Graphic Elements."
- **Properties Editor.** This editor shows the properties belonging to one or more currently selected elements.
- **Animation Summary.** This area 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 Symbol Editor to create symbols containing multiple visual and functional configurations called Symbol Wizards. For more information, see, "Creating Multiple Configurations of a Symbol" on page 84.

Tools Panel

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

The Tools panel includes:

- Basic graphic elements such as lines, rectangles, polygons, arcs, and text.
- 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 selected ArcestrA attributes.

For more conceptual information, see "Elements" on page 25.

For more information on how to use elements, see "Working with Graphic Elements" on page 109.

Elements List

The Elements area shows 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. You can use the Elements List to:

- See a list of all elements, groups of elements, and embedded symbols 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 367.

Properties Editor

You can 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 30.

For more information on how to use element properties, see "Editing Common Properties of Elements and Symbols" on page 163.

Animations Summary

You can use the Animations summary to review, select, and configure the animation behavior of an element selected on the canvas.

For an overview of the different animation types, see "Animation Types" on page 35.

For more information on how to use the animations, see "Animating Graphic Elements" on page 269.

Canvas

The canvas is your drawing area. You use it as you would in other image editing software by drawing elements and changing them to your requirements.

Elements

You use elements to create a symbol. The ArchestrA Symbol 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

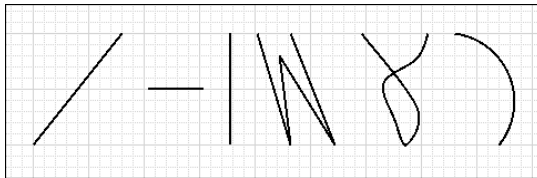
You can embed the following on the canvas:

- Imported Client Controls
- Other symbols

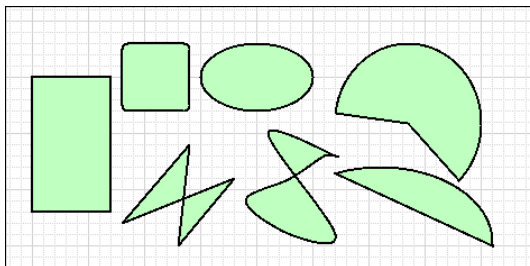
Basic Elements

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

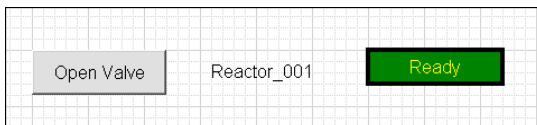
- Open elements, such as lines, H/V lines, polylines, curves, and arcs.



- 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.



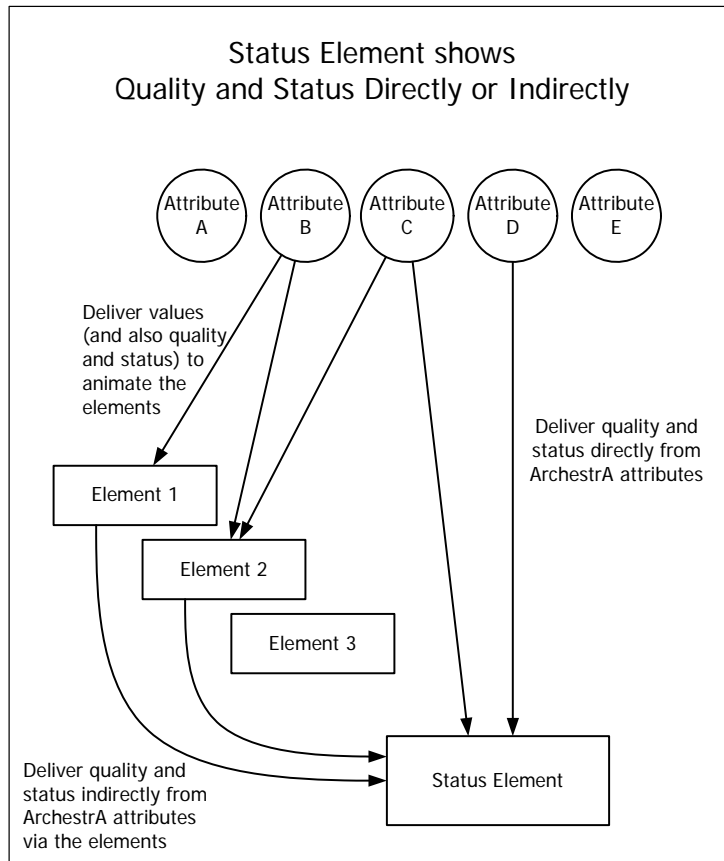
- Text elements, such as buttons, text, and text boxes.



Status Element

You can use a status element to monitor and indicate the status or quality of:

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



Abnormal quality and status can be:











- An error caused by communication, configuration, operational, security, or software errors.
- Bad, initializing, pending, uncertain, or warning quality.

The status element enables you to monitor:

- One or more elements on the canvas that contain animations.
- One or more ArchestrA attributes, values, or expressions.

A status element can monitor multiple attributes at the same time, but only show one icon. A status element always shows the most severe quality or status.

A status element prioritizes error and status conditions, as shown in the following list

| | |
|---|--|
|  | Communication Error (highest priority) |
|  | Configuration Error |
|  | Pending |
|  | Operational Error |
|  | Software Error |
|  | Security Error |
|  | Warning |
|  | Bad |
|  | Uncertain |
|  | Initializing (lowest priority) |

For more information, see "Configuring Animation for a Status Element" on page 341.

You can also change the icons that are shown for each status or quality. For more information, see "Showing Quality and Status" on page 53.

Windows Common Controls

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

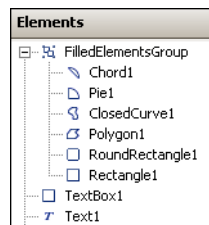
- A radio button group to select an option from a mutually exclusive group of options.
- A check box to add a selectable option.
- An edit box to add an entry box for text.
- A combo box to select an option from a drop-down list.
- A calendar to use a date selection control.

- 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.



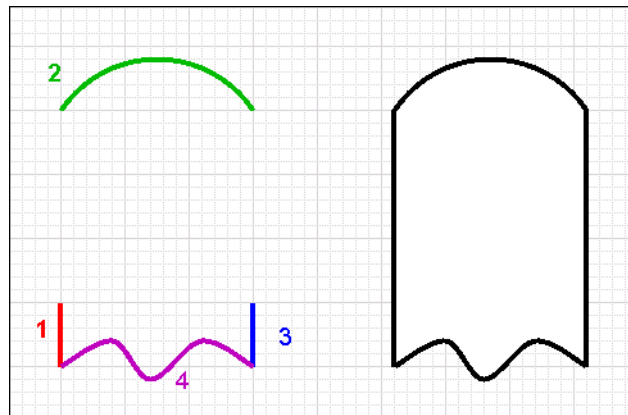
For example, you can create a series of elements that model a valve in your facility. When the valve has all the properties and animations you want, you can group the elements together.

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, to create 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 from top to bottom.
- 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 ArcestrA symbol to extend its functionality.

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

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

You can embed a symbol that contains an embedded client control into a managed InTouch application and use the functionality of the client control directly in the InTouch HMI.

For more information, see "Using Client Controls" on page 387.

Properties

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

There are two types of properties:

- Predefined properties
- Custom properties

When you configure an element to reference one of its **own** properties in a configuration field or a script, you can just use its property name. For ArcestrA symbols, there are no self-referencing keywords such as "me." as used for AutomationObjects.

You can, however, use the "me." keyword to reference attributes of the AutomationObject that is hosting the ArchestrA symbol you are currently configuring.

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

You can use custom properties to extend the functionality of a symbol. A custom property can contain:

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

For example, for a tank symbol 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 253.

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 Percentage (Transparency)
- Locked (Locked)
- Enablement (Enabled)
- Tab Order (TabOrder)
- Tab Stop (TabStop)
- Single Object Treatment (TreatAsIcon)
- Visibility (Visible)

Changing a Group Name

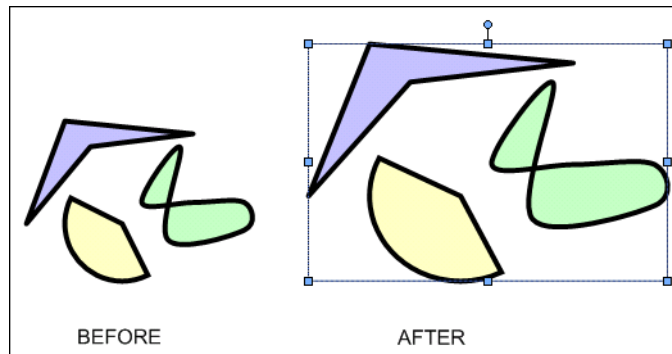
If you change the group name, it has no affect on the contained elements. The contained elements keep their name.

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.

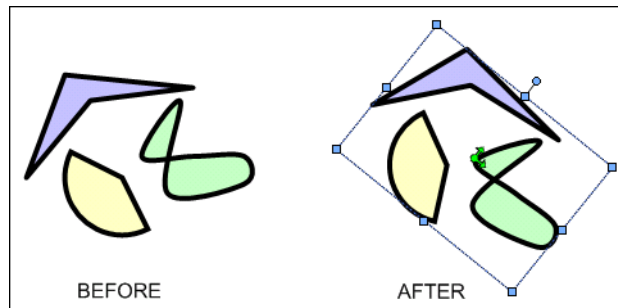
Changing the Size of a Group

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



Changing the Orientation of a Group

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



Changing the Transparency of a Group

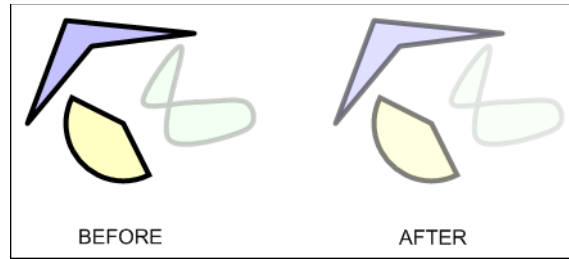
If you increase the transparency of the group, all contained objects appear more transparent, but their own transparency property values do not change. If you change their transparency values, it is in relation to the transparency level of the group.

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 affect 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 367.

Animations

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

For example, you can bind the vertical fill of a rectangle to an ArchestrA 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:

| Animation Type | Description |
|--------------------------|--|
| Visibility | Shows or hides the element depending on a value or an expression. |
| Fill Style | Specifies the interior fill style depending on a discrete or analog expression or one or more conditions. |
| Line Style | Specifies the style and pattern of the element line depending on a discrete or analog expression or one or more conditions. |
| Text Style | Specifies the style of the element text depending on a discrete or analog expression or one or more conditions. |
| Blink | Sets the element to blink invisibly or with specified colors depending on a discrete value or expression. |
| Element Style | Defines a set of visual properties that determine the appearance of text, lines, graphic outlines, and interior fill shown in ArchestrA Symbols or graphics. |
| Alarm Border | Shows a colored border around a symbol that represents the current alarm state of equipment represented by a symbol. |
| % Fill Horizontal | Fills the element with color partially from left to right or vice versa, depending on an analog value or expression. |
| % Fill Vertical | Fills the element with color partially from top to bottom or vice versa, depending on an analog value or expression. |

| Animation Type | Description |
|----------------------------|---|
| Location Horizontal | Positions the element with a horizontal offset depending on an analog value or expression. |
| Location Vertical | Positions the element with a vertical offset depending on an analog value or expression. |
| Width | Increases or decreases the element width depending on an analog value or expression. |
| Height | Increases or decreases the element height depending on an analog value or expression. |
| Point | Changes the X and Y coordinate values of one or more selected points on a symbol or graphic element. |
| Orientation | Rotates the element by an angle around its center point or any other point depending on an analog value or expression. |
| Value Display | Shows a discrete, analog, string value, time value, name or expression. |
| Tooltip | Shows a value or expression as a tooltip when the mouse is moved over the element. |
| Disable | Disables the element's animation depending on a Boolean value or expression. |
| User Input | Allows the run-time user to type a Boolean, analog, string, time or elapsed time value that is then assigned to an attribute. |
| Slider Horizontal | Allows the run-time user to drag the element left or right and write back the offset to an analog attribute. |
| Slider Vertical | Allows the run-time user to drag the element up or down and write back the offset to an analog attribute. |
| Pushbutton | Writes predetermined values to Boolean or analog references when the user clicks on the element. |
| Action Scripts | Runs an action script when the run-time user clicks on the element. |

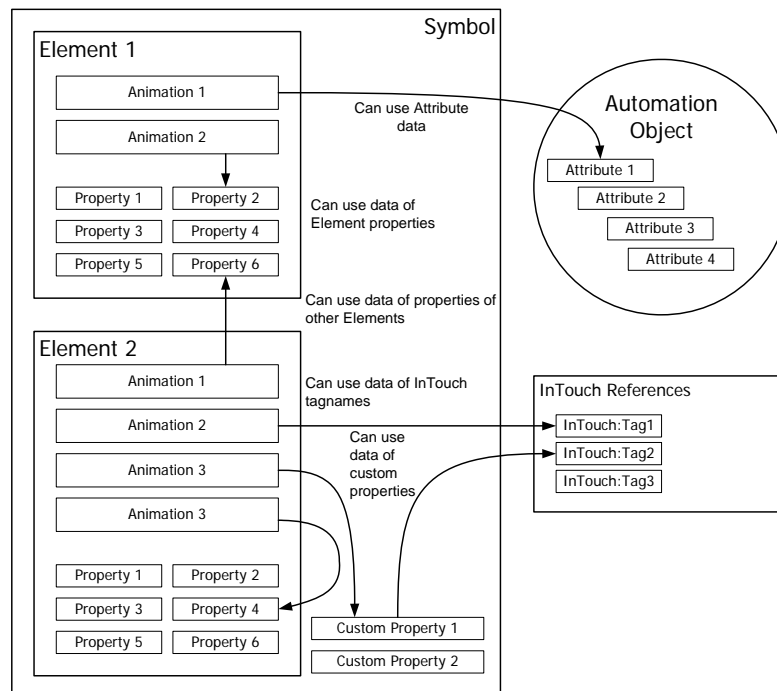
| Animation Type | Description |
|--------------------|--|
| Show Symbol | Shows a specified symbol at a specified position when the run-time user clicks on the element. |
| Hide Symbol | Hides a specified symbol when the run-time user clicks on the element. |

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 symbol.
- Custom properties of a symbol.
- InTouch tags.

To use the data of an InTouch tag in an ArchestrA symbol, you need to use the special InTouch keyword in the configuration field. The syntax is: `InTouch:TagName`, where *TagName* is a tag that you intend using in the InTouch application into which you are embedding the ArchestrA symbol.



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, you can 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 that is specific to the type of user input used for animation.

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.

Embedded Symbols

You can embed symbols from the Graphic Toolbox, AutomationObject templates, and instances into other symbols. Embedding symbols enables you to rapidly develop more complex symbols with common components.

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

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

Appearance of Embedded Symbols

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

Changing Embedded Symbols

After you embed a symbol, you can change its size, orientation or transparency. You can add a limited set of animations to the symbol, 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 symbol from within the hosting symbol.
- Embed a symbol contained in an `AutomationObject` into a symbol that is contained in the Graphic Toolbox.
- Create circular references. A circular reference occurs when one symbol (Symbol A) has embedded within it another symbol (Symbol B) that has embedded within it a symbol that directly or indirect references back to the first symbol (Symbol A).

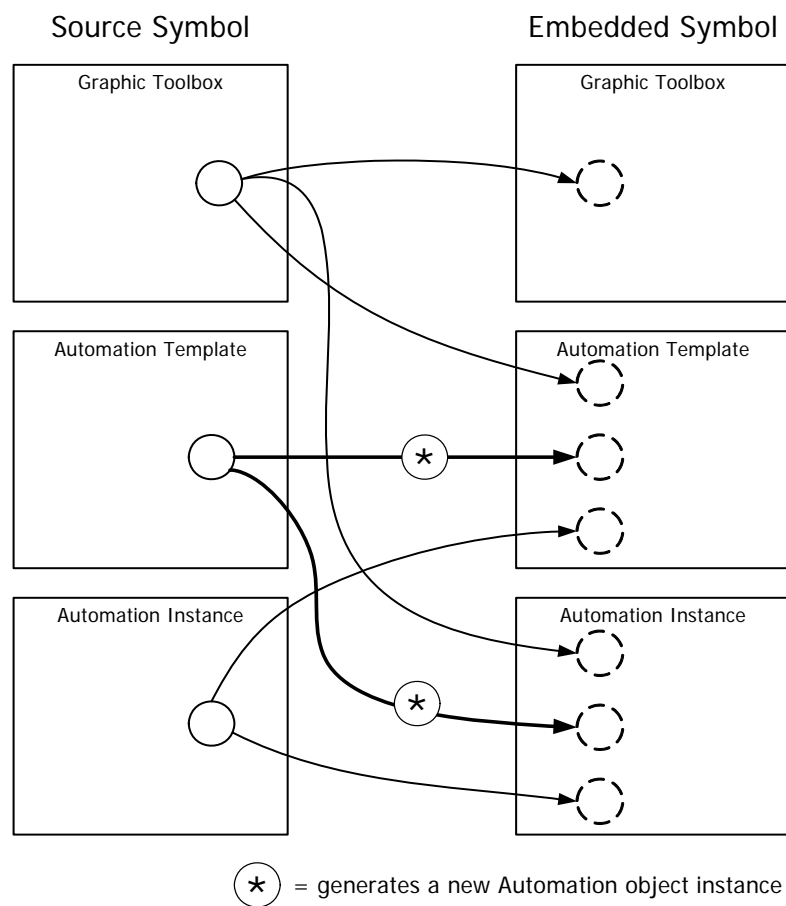
You can, however, change the embedded symbol by changing its source symbol. The changes you make propagate to the embedded symbol.

Embedding and Instantiation

You can embed symbols into symbols contained in the Graphic Toolbox, an AutomationObject template, or an AutomationObject instance.

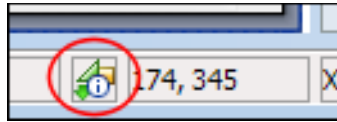
You can embed symbols contained in an AutomationObject template into symbols contained only in other AutomationObjects. When you do so, a new AutomationObject instance is created. You can give it a name, and the new instance inherits the symbol, but does not contain it.

You can only embed symbols contained in an AutomationObject instance into symbols contained in other AutomationObjects. The template or instance inherits the symbol, but does not contain it.



Symbol Change Propagation

When you make changes and save a symbol, the changes are propagated to all other symbols that embed that symbol. The ArcestrA Symbol Editor shows an icon in the status bar beneath the canvas that indicates that the source of an embedded symbol changed.



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

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

When a symbol is changed, its external size can also be changed. ArcestrA symbols support dynamic size propagation and anchor points that let you determine how and if size changes are propagated. For more information about size propagation, see "Size Propagation and Anchor Points" on page 43.

If the symbol is hosted by the Graphic Toolbox and edited:

- All symbols hosted by AutomationObject templates and instances that contain embedded instances of this symbol are also updated.
- All embedded instances of this symbol in InTouch WindowMaker are also marked for an update.

If the symbol is hosted by an AutomationObject and edited:

- All symbols hosted by derived AutomationObjects are also updated.
- All embedded instances of this symbol in InTouch WindowMaker hosted by derived AutomationObjects are marked for an update.

Size Propagation and Anchor Points

To control how symbol size changes are propagated to embedded instances, the symbol uses an anchor point. By default, the anchor point of the symbol 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 symbol center.

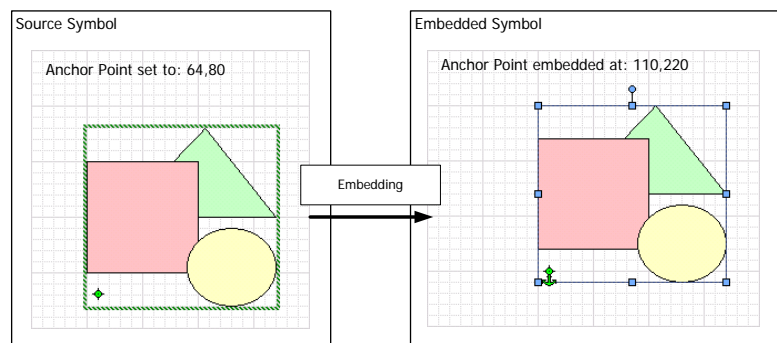
When you embed a symbol, the embedded symbol inherits the anchor point in relation to its own center point.

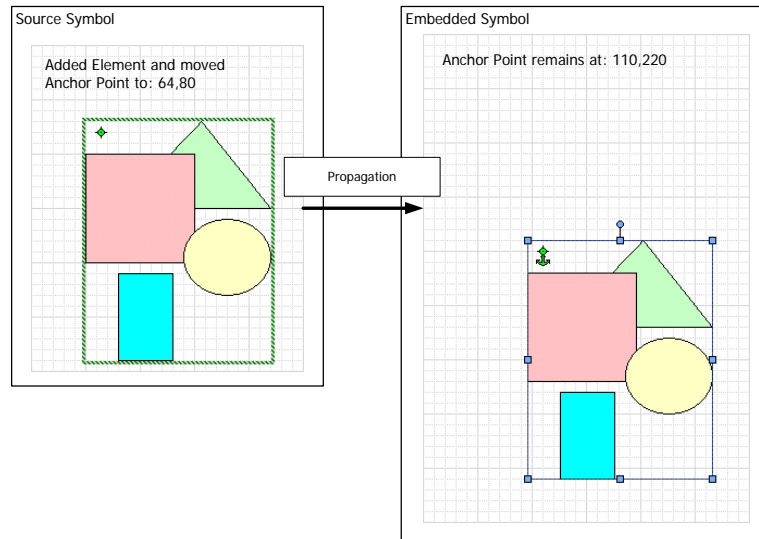
You can also set the `AnchorFixedTo` property. When you make changes to the symbol that affects its size, the `AnchorFixedTo` property determines if the absolute position or relative position of the anchor point is recalculated. This property can have following values:

- **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 symbol. This affects the position of the embedded instances. The anchor points of the embedded instances, however, remain unchanged.





You can change the anchor point of an embedded symbol. This moves the embedded symbol. It does not change the anchor point position in relation to the symbol.

You can resize or rotate the embedded symbol. The anchor point moves in relation to the embedded symbol.

Note: You can also use the AnchorPoint property in the Properties Editor to change the position.

Estimating Graphic Performance

You can gauge the performance of an ArcestrA symbol at run time using the Graphics Performance Index (GPI). The GPI calculates the estimated call up time when the symbol you are building in the Symbol 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 symbol launched in the InTouch WindowViewer at run time.

Estimating Symbol Performance

Use the Graphics Performance Index (GPI) window to view estimated performance statistics of a symbol 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 symbols currently in a non-editable state.

To estimate symbol performance using the Graphics Performance Index

- 1 Do any of the following in the ArcestrA Symbol 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 symbol. A rating of 5 indicates a symbol call up time less than 1 second. See "Understanding GPI Rating Calculations" on page 46 for details about how the GPI rating is calculated.

Important: 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.

- 2 Click the **Details** button to expand the window. A list displays showing rows of details for component types in the symbol 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.

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

Understanding GPI Rating Calculations

GPI rating calculations are based on components in a symbol displayed in WindowViewer running on a computer with an Intel Core 2 Duo CPU and 4 GB of RAM.

All visible graphics on the screen are counted, except for Symbol Wizard symbols that may be stored in memory, which reduces the amount of content loaded and rendered at run time. Exceptions to Symbol Wizard symbol 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 symbols are not counted, if any graphic element in a Symbol Wizard symbol 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 |

| Element Counter Type | Counter Description | Number of Items | Impact Score |
|-----------------------------|----------------------------|------------------------|---------------------|
| Text | Number of Texts | 50000 | 11575.6 |
| TextBox | Number of Text Boxes | 50000 | 5526.2 |
| 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:

| Animation Counter Type | Counter Description | Number of Items | Impact Score |
|-------------------------------|--|------------------------|---------------------|
| 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 |

| Animation Counter Type | Counter Description | Number of Items | Impact Score |
|-------------------------------|--|------------------------|---------------------|
| 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 symbols are not counted, if any named custom property in a Symbol Wizard symbol 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 |
| 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 symbols are not counted, if any named script in a Symbol Wizard symbol 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 Symbol with a 4.5 GPI Rating

The following table shows values for components in a sample symbol 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 |
| | Number of Groups | 20 | 400 | 0.05 |
| | Number of Embedded Symbols | 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 |
| 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 |

| Category | Performance Counter | Config. Count | Processing Capacity per second | Projected Time (sec.) |
|----------------|--|---------------|--------------------------------|-----------------------|
| 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 |
| | | | Total | 1.70 |
| | | | GPI | 4.50 |

Saving a Symbol that May Impact Run-time Performance

When building a symbol or attempting to save a symbol 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 symbol 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 103.

Showing Quality and Status

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

- Use a Status element that shows you an icon. It indicates the status or quality of specified ArcestrA attributes directly or those used indirectly in elements.
- Change the appearance of animated elements based on the status and quality of ArcestrA attributes 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 341.

For more information on how to configure the appearance of a status element, see "Setting the Appearance of a Status Element" on page 222.

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.

This also applies to:

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

This 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" on page 224.

Chapter 2

Comparing WindowMaker and ArcestrA Symbol Editor

You can use the ArcestrA Symbol Editor to do most of the tasks you do in InTouch WindowMaker. You can also use many of the same shortcut keys.

Differences between WindowMaker and the ArcestrA Symbol Editor

The ArcestrA Symbol Editor has features that are not available in InTouch WindowMaker, such as:

- Additional elements.
- Additional and enhanced appearance of the elements.
- Additional and enhanced design-time functionality.

Elements

Elements are the graphical objects you use to create an ArcestrA Symbol. The ArcestrA Symbol Editor provides elements that are not available in InTouch WindowMaker, such as:

- Curves and closed curves.
- Arcs, pies, and chords defined by two or three points.
- Status elements to conditionally show an icon depending on quality and status of attribute data.

- Path graphics that you create by joining line-based elements together to form a new closed element.
- Windows common controls, such as the Calendar control and Date Time Picker control.

Appearance

The ArcestrA Symbol Editor extends the InTouch graphic configuration. For example, you can use:

- Gradients for line, fill, and text color.
- Patterns for line, fill, and text color.
- Textures for line, fill, and text color.
- Partial transparency.
- Fill behavior in relation to a symbol or screen.

Enhanced Functionality

The ArcestrA Symbol Editor provides a entire array of enhancements to make your life easier when creating visualization for your manufacturing environment.

Usability Enhancements

The ArcestrA Symbol Editor makes it easy to select and configure elements. You can:

- Select elements from a list as well as from the canvas. This lets you select elements beneath others without having to move them.
- View and change properties and animation (links) of an element by simply selecting it on the canvas.
- Edit elements contained in groups and path graphics without having to break the group or path graphic. This is called inline editing.

Style Replication

Using the Format Painter, you can simply apply the style of one element with one click to another element, even to an element of a different type.

Animation Replication

Using the ArcestrA Symbol Editor you can copy, cut, and paste animations from one element to another element, even to an element of a different type.

Element Positioning

The ArcestrA Symbol Editor extends the positioning feature of InTouch WindowMaker and lets you:

- Distribute elements equally in horizontal or vertical direction.
- Make elements same horizontal and/or vertical size.
- Increase or decrease horizontal or vertical space.
- Remove horizontal or vertical space between elements.
- Lock an element so that you do not accidentally move or edit it.
- Rotate any element at design time by any angle around a center of rotation.
- Apply resizing and rotating to multiple elements at the same time.
- Move the z-order of an element one level backward or forward.
- Align text within text boxes and buttons.

Group Functionality

The ArcestrA Symbol Editor uses the concept of groups instead of the cell and symbol concepts of InTouch WindowMaker. You can:

- Embed groups within groups.
- Edit individual elements within a group (or an embedded group) without breaking up the group.
- Easily remove elements from or add elements to existing groups.

Extensibility with Custom Properties

You can add custom properties to a symbol or embedded symbol. You can connect custom properties to AutomationObject attributes, element properties, and even InTouch tags. You can use the custom properties as you would with any pre-defined property at design time and run time.

Note: ArcestrA custom properties referencing InTouch tags which have hyphens in their names will not work in run time. For example, "InTouch: TAG-1" will not work in run time.

Element Styles

Element Styles define one or more of the fill, line, text, blink, outline, and status properties of graphic elements. Apply an Element Style to a graphic element to set the element to the preconfigured properties defined in that Element Style. The element's local properties that are defined in the Element Style are disabled.

Element Styles help drive standards for screen builders and others who are creating symbols.

Miscellaneous Enhancements

Using the ArcestrA Symbol Editor, you can:

- Access the properties of the elements and custom properties of the symbol through scripting.
- Set the tab order of the elements.
- Use line end styles, such as arrows.
- Dynamically disable specific animations from elements without losing the configuration information.
- Use image meta files and other image formats.
- Use anti-aliasing to improve how the symbol is shown.

Procedures for Common WindowMaker Tasks and Techniques

Most of the configuration that you do in InTouch WindowMaker can be easily done in the ArcestrA Symbol Editor. There are some differences between and similarities of graphics, animations, and scripts.

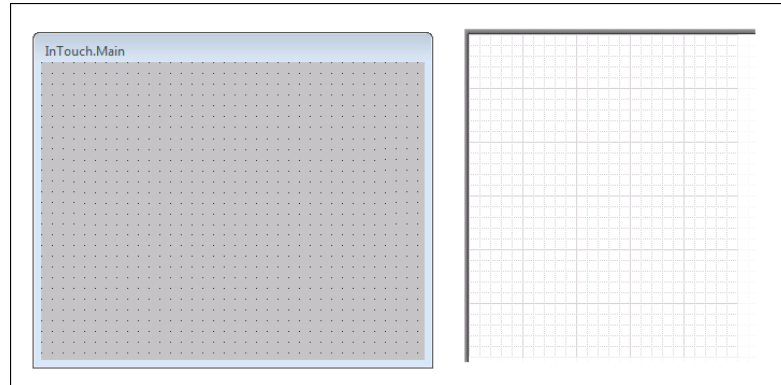
Using Graphics

You can use the ArcestrA Symbol Editor in basically the same way as you use InTouch WindowMaker. The ArcestrA Symbol Editor includes a drawing area on which you can place graphical objects to construct a visual representation of production processes and to provide an interface between a human and a machine.

Some objects you use in InTouch do not exist in the ArcestrA Symbol Editor, such as ActiveX controls and some Wizards. Their functionality is replaced other controls that are more powerful and integrate better into the ArcestrA environment.

Using the Drawing Area

The drawing area of the ArcestrA Symbol Editor is called the canvas. You use it like an InTouch window. The maximum size of the canvas is 2,000 by 2,000 pixels.



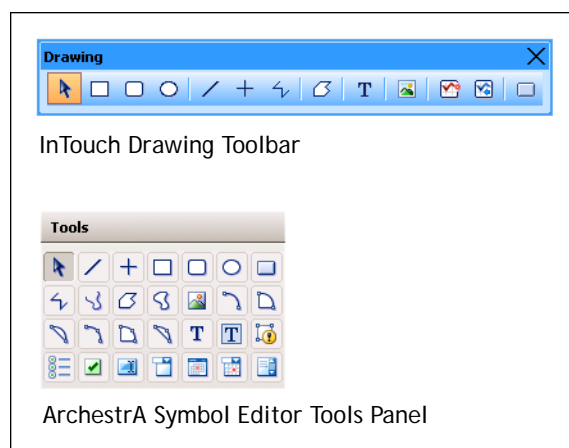
Setting the Drawing Area Color

You cannot set the drawing area color in the ArcestrA Symbol Editor. The drawing area color is transparent and inherits the color of the environment that the symbol is embedded into.

If you embed an ArcestrA Symbol into an InTouch window, the area between the elements adopts the color of the InTouch window.

Using Basic Objects

InTouch basic objects such as rectangles, ellipses, and polylines can be drawn in very similar way in the ArcestrA Symbol Editor. The basic objects are called elements in the ArcestrA Symbol Editor.



Using Complex Objects

InTouch objects such as ActiveX controls, Wizards, cells, and symbols do not exist in the ArcestrA Symbol Editor.

You can, however, import SmartSymbols into an ArcestrA symbol. When you import a SmartSymbol, the elements and animations of a SmartSymbol are converted.

In the ArcestrA Symbol Editor, you can create groups of elements. Groups maintain the properties of the contained individual elements. You can set the TreatAsIcon property of a group to change the behavior of the group.

Using Wizards

You cannot import InTouch Wizards to an ArcestrA Symbol or into the Graphic Toolbox. Instead, use:

- The ArcestrA Symbol Library, which you can import into the Graphic Toolbox.
- Windows controls that are part of the Toolbox. You can use:
 - Radio button groups
 - Check boxes
 - Edit boxes
 - Combo boxes
 - Calendar control
 - DateTime picker
 - List boxes

Using Animations

You can use animations in the ArcestrA Symbol Editor to set run-time behavior of the symbols as you would in InTouch WindowMaker. You can configure one or more animations for an element or symbol. The data can come from various sources.

Configuring Data Sources

In InTouch WindowMaker, you use the Tagname Dictionary to define variables that hold values. In the ArcestrA Symbol Editor, data sources can be:

- ArcestrA AutomationObject attributes.
- Custom properties and inherited properties of the symbol.
- InTouch tags themselves. The ArcestrA Symbol Editor uses a special InTouch reference you can use to directly connect to InTouch tags.

Using Data Types

ArcestrA Symbols use the ArcestrA data types, which are different than InTouch data types.

The following table shows you the data types of both and how they correspond to each other:

| InTouch | ArcestrA | Description |
|----------|-----------------|--|
| Discrete | Boolean | Boolean value. For example: 0 or 1 |
| Integer | Integer | Integer value. For example: -4, 7, or 22 |
| Real | Float or Double | Float or double value with different precision. For example: 3.141, -5.332, or 1.343e+17 |
| Message | String | String value. For example: "Hello World" |
| n/a | DateTime | Datetime value. For example: "04/13/2006 04:03:22.222 AM" |

| InTouch | ArcestrA | Description |
|---------|-------------------------|---|
| n/a | ElapsedTime | <p>Float value that represents a time that has elapsed in seconds. It is shown often in the following format, but is stored as a float value.</p> <p>[-][DDDDDD] [HH:MM:]SS[.ffffff]</p> <p>Values are as follows:</p> <ul style="list-style-type: none"> • DDDDDD is from 0 to 999999 • HH is from 0 to 23 • MM is from 0 to 59 • SS is from 0 to 59 • fffffff is fractional seconds to right of the decimal <p>Elapsed time can be positive or negative.</p> |
| n/a | InternationalizedString | A special string data type that can store special characters. |

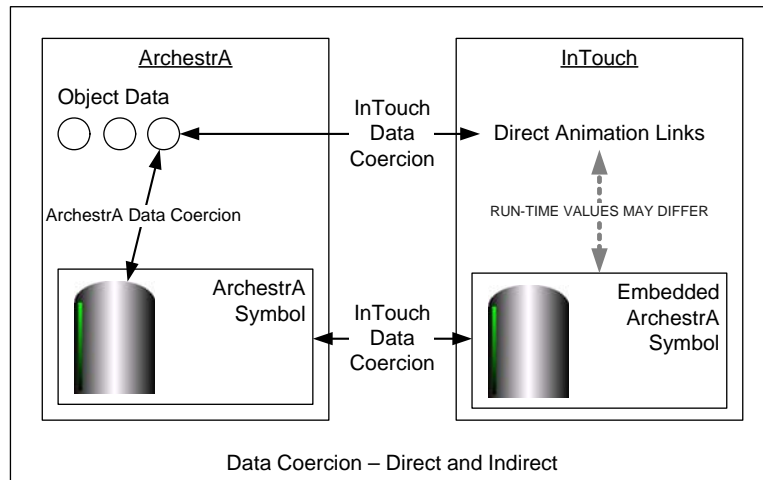
You can configure ArcestrA Symbols to retrieve data from the Galaxy.

When the source data type is different than the data type it is used for, the data is coerced according to the rules of ArcestrA data coercion and a string value of "-10" is coerced to "True" in the animation.

If you embed this ArcestrA Symbol into an InTouch window, the data type of the animation link is coerced according to the InTouch data coercion. The embedded ArcestrA Symbol shows "True" in the InTouch HMI.

However, if you directly create an discrete animation display link in the InTouch HMI that points at the original data source, the resulting value can be different.

In this example the string value "-10" is shown as "False" in the InTouch HMI.



Using Animations

You configure InTouch animations using the **Animation Links** dialog box. You can open this dialog box by double-clicking on an InTouch object.

You configure animations in the ArcestrA Symbol Editor using the **Edit Animations** dialog box, which is normally opened by double-clicking on an element.

Some of the animation types are different and others have been grouped to simplify configuration. Use the following table to find the equivalent animation type in the ArcestrA Symbol Editor:

| InTouch Animation | ArcestrA Symbol Editor Animation |
|------------------------------------|---|
| User Inputs - Discrete | User Input - Boolean |
| User Inputs - Analog | User Input - Analog |
| User Inputs - String | User Input - String |
| Sliders - Vertical | Slider Vertical |
| Sliders - Horizontal | Slider Horizontal |
| Touch Pushbuttons - Discrete Value | Pushbutton - Boolean |
| Action | Action Scripts |
| Show Window | (not supported) |
| Hide Window | (not supported) |
| Line Color - Discrete | Line Style - Boolean |

| InTouch Animation | ArcestrA Symbol Editor Animation |
|-----------------------------|---|
| Line Color - Analog | Line Style - Truth Table |
| Line Color - Discrete Alarm | converted to Line Style |
| Line Color - Analog Alarm | converted to Line Style |
| Fill Color - Discrete | Fill Style - Boolean |
| Fill Color - Analog | Fill Style - Truth Table |
| Fill Color - Discrete Alarm | converted to Fill Style |
| Fill Color - Analog Alarm | converted to Fill Style |
| Text Color - Discrete | Text Style - Boolean |
| Text Color - Analog | Text Style - Truth Table |
| Text Color - Discrete Alarm | converted to Text Style |
| Text Color - Analog Alarm | converted to Text Style |
| Object Size - Height | Height |
| Object Size - Width | Width |
| Location - Vertical | Location Vertical |
| Location - Horizontal | Location Horizontal |
| Percent Fill - Vertical | % Fill Vertical |
| Percent Fill - Horizontal | % Fill Horizontal |
| Miscellaneous - Visibility | Visibility |
| Miscellaneous - Blink | Blink |
| Miscellaneous - Orientation | Orientation |
| Miscellaneous - Disable | Disable |
| Miscellaneous - Tooltip | Tooltip |
| Value Display - Discrete | Value Display - Boolean |
| Value Display - Analog | Value Display - Analog |
| Value Display - String | Value Display - String |

Using Scripts

You can configure scripts in ArcestrA Symbol Editor the same way as you do in InTouch WindowMaker. There are, however, some small differences:

| InTouch Script | ArcestrA Symbol Editor Script |
|-----------------------|--|
| Application Script | (not available) |
| Window Script | Symbol Predefined Script |
| Key Script | Action Script animation with a key trigger |
| Condition Script | Symbol Named Script with an OnTrue, OnFalse, WhileTrue or WhileFalse trigger |
| Data Change Script | Symbol Named Script with a DataChange trigger |
| QuickFunction | (not available) |
| ActiveX Event Script | (not available) |
| Action Script | Action Script animation |

Using Application Scripts

In the InTouch HMI, application scripts are triggered:

- One time when the application starts up in WindowViewer.
- Periodically when the application runs in WindowViewer.
- One time when the application shuts down in WindowViewer.

ArcestrA Symbols correspond to InTouch applications and enable you to configure predefined scripts directly associated with a symbol. Such are:

- On Show
- While Showing
- On Hide

Using Key Scripts

You cannot use key scripts in the ArcestrA Symbol Editor, but you can associate an element with an action script that is activated by a key combination.

Using Condition Scripts

You can configure a script that runs when a condition is fulfilled by using the Symbol Scripts feature. It lets you define triggers that run a script when a value or expression:

- Is fulfilled. (WhileTrue)
- Becomes fulfilled. (OnTrue)
- Is not fulfilled. (WhileFalse)
- Becomes no longer fulfilled. (OnFalse)

Using Data Change Scripts

You can configure a script that is run when a value or expression changes by using the Symbol Scripts feature. It lets you define a trigger that runs a script when a value or expression changes.

Using Action Scripts

You can configure Action Scripts in the ArcestrA Symbol Editor in the same way as you would in InTouch WindowMaker. When the run-time user interacts with an element, such as with the mouse or by pressing a key, an action script can be triggered to run.

You use the InTouch action script window to create action scripts. You use the ArcestrA Symbol Editor action script window to create action scripts.

You can configure action scripts for individual elements or for the entire symbol.

You can use many of the predefined functions of InTouch WindowMaker in the ArcestrA Symbol Editor. For a complete list of InTouch predefined functions that can be used with ArcestrA Symbols, see "Importing Action Scripts" on page 422.

Other InTouch script types, such as application scripts and key scripts, can be configured with ArcestrA AutomationObjects.

Chapter 3

Managing Symbols

This section describes ArcestrA symbols, how they are stored in the ArcestrA environment, and how they are managed from the IDE.

About Symbols

ArcestrA symbols are graphical symbols you use to visualize data in an InTouch application.

You manage ArcestrA symbols from the IDE to:

- Create a new symbol.
- Edit a symbol with the ArcestrA Symbol Editor.
- Organize symbols within the Graphic Toolbox.
- Duplicate symbols.
- Import and export symbols.
- Delete a symbol.
- Configure security for a symbol's operations.
- Open the symbol in read-only mode with the ArcestrA Symbol Editor.

Creating a New Symbol

You can create a new symbol from:

- The Graphic Toolbox for generic symbols that you frequently use in different situations. For example, a valve symbol.
- The **Graphics** tab of an AutomationObject template. Do this if you want to re-use the symbol in combination with the object functionality. An example is a symbol representing a specific tank and your production facility has multiple tanks.
- The **Graphics** tab of an AutomationObject instance. Do this if you are unlikely to re-use the symbol in any other situation.

Creating Symbols in the Graphic Toolbox

You can create a new symbol in the Graphic Toolbox, which is then listed in the Graphic Toolbox with a default name. You can:

- Rename the symbol.
- Move the symbol.
- Edit the symbol with the ArcestrA Symbol Editor.

To create a new symbol from the IDE

- 1 On the **Galaxy** menu, point to **New**, and then click **Symbol**. The Graphic Toolbox appears and a new symbol is listed.

Note: You can also press **Ctrl + Shift + S** to create a new ArcestrA Symbol or right-click and then select **New** and **Symbol** from the shortcut menu.

- 2 Rename the symbol.

Names must be unique within the entire Graphic Toolbox hierarchy. Valid characters for symbol names include alphanumeric characters, #, and _ (underscore). Symbol names cannot contain spaces and the symbol name cannot begin with the \$character.

- 3 Double-click the symbol name. The ArcestrA Symbol Editor appears.
- 4 Draw the symbol. For specific information about using the drawing tools, see "Working with Graphic Elements" on page 109.

Creating Symbols in AutomationObject Templates

You can create a symbol from the **Graphics** tab in an AutomationObject template. Creating a symbol this way automatically associates the new symbol with the AutomationObject.

To create a new symbol for an AutomationObject template

- 1 Open the AutomationObject template. Click the **Graphics** tab. Any local and inherited symbols are listed.
- 2 Click the **New Symbol** icon and assign a name to the new symbol.
Names must be unique. Valid characters for symbol names include alphanumeric characters, \$, #, and _ (underscore). Symbol names cannot include spaces and the symbol name cannot begin with the \$ character.
- 3 If needed, type the description of the symbol in the **Description** box.
- 4 Click the symbol name and click **Open**. The ArcestrA Symbol Editor appears.
- 5 Draw the symbol. For specific information about using the drawing tools, see "Working with Graphic Elements" on page 109.

Creating Symbols in AutomationObject Instances

You can create a symbol from the **Graphics** tab in an AutomationObject instance. Creating a symbol this way automatically associates the new symbol with the AutomationObject instance.

Note: AutomationObjects can also inherit symbols from their parent template. You can only view an inherited graphic in read-only mode. Inherited graphics cannot be removed or edited.

To create a new symbol for an AutomationObject instance

- 1 Open the AutomationObject instance and click the **Graphics** tab.
Any local and inherited symbols are listed.
- 2 Click the **New Symbol** icon. Give the new symbol a name.
Names must be unique. Valid characters for symbol names include alphanumeric characters, \$, #, and _ (underscore). Symbol names cannot include spaces and the symbol name cannot begin with the \$ character.
- 3 If needed, type the description of the symbol in the **Description** box.
- 4 Select the symbol name, and then click **Open**. The ArcestrA Symbol Editor appears.

- 5 Draw the symbol. For specific information about using the drawing tools, see "Working with Graphic Elements" on page 109.

Opening Symbols for Editing

You can start the ArcestrA Symbol Editor from a symbol:

- Contained in the Graphic Toolbox.
- Contained in an AutomationObject template or instance.
- Embedded in an InTouch window.

You check out a symbol when you open it for editing with the ArcestrA Symbol Editor. No other user can edit the symbol while you have it checked out.

You can open multiple instances of the ArcestrA Symbol Editor at the same time. However, you cannot edit the same symbol in multiple instances of the ArcestrA Symbol Editor.

To edit a symbol in the Graphic Toolbox

- 1 Open the Graphic Toolbox.
- 2 Browse to the symbol you want to edit.
- 3 Double-click the symbol. The ArcestrA Symbol Editor appears.
- 4 Edit the symbol. For specific information about using the drawing tools, see "Working with Graphic Elements" on page 109.
- 5 Click **Save and Close**. The ArcestrA Symbol Editor closes and the updated symbol is checked in.

To edit a symbol contained in an AutomationObject

- 1 Open the AutomationObject.
- 2 Click the **Graphics** tab.
- 3 Select the symbol to edit and click **Open**. The ArcestrA Symbol Editor appears.
- 4 Edit the symbol.
 - a Embed a symbol or use a graphic tool from the Tools menu to add a graphic element to your symbol.
 - b If you want to modify a symbol you embedded, right-click on the symbol to show the symbol's shortcut menu. Then, select Embedded Symbol followed by Edit Symbol.

For specific information about using the drawing tools, see "Working with Graphic Elements" on page 109.

- 5 Click **Save and Close**. The ArcestrA Symbol Editor closes and the updated symbol is checked in. Depending on the AutomationObject, a confirmation message may appear. Click **Yes** to save.

To edit a symbol that is embedded into an InTouch window

- 1 In WindowMaker, open the InTouch window that contains the embedded symbol.
- 2 Right-click the embedded symbol to edit, point to **ArcestrA Graphic "Symbolname"**, and then click **Edit Symbol**. The ArcestrA Symbol Editor appears.
- 3 Edit the symbol. For specific information about using the drawing tools, see "Working with Graphic Elements" on page 109.
- 4 Click **Save and Close**. The ArcestrA Symbol Editor is closed and the updated symbol is checked in.

Note: To leave the symbol checked out, click **Keep Checked Out** in the ArcestrA Symbol Editor. This ensures that no other user can check out your symbol for editing.

Organizing Symbols in the Graphic Toolbox

You use the Graphic Toolbox to organize your symbols by creating a folder hierarchy as you would with files and folders in Windows Explorer. You can move symbols around within the folder hierarchy. These folders are called Graphic Toolsets.

Creating Graphic Toolsets in the Graphic Toolbox

You can create Graphic Toolsets in the Graphic Toolbox to organize your symbols. For example you can create a Graphic Toolset called "Valves" to store different valve symbols.

To create a Graphic Toolset in the Toolbox

- 1 Open the Graphic Toolbox.
- 2 Select the Graphic Toolset under which you want to create a new Graphic Toolset. Select the Galaxy name if there are currently no Graphic Toolsets.
- 3 On the **Galaxy** menu, point to **New**, and then click **Graphic Toolset**. A new Graphic Toolset is created with a default name.
- 4 Rename the new Graphic Toolset as needed.

Moving Symbols between Graphic Toolsets

You can move symbols from one Graphic Toolset in the Graphic Toolbox to another. Moving symbols between Graphic Toolsets does not affect its functionality.

To move symbols between Graphic Toolsets in the Graphic Toolbox

- 1 Open the Graphic Toolbox.
- 2 Locate the symbol you want to move.
- 3 Do either of the following:
 - Drag the symbol to the Graphic Toolset you want to place it in. The symbol moves to the new Graphic Toolset.
 - To move the symbol to the top level in the Graphic Toolset hierarchy, drag the symbol to the Galaxy name icon.

Renaming Symbols

You can rename a symbol at any time. Renaming a symbol does not affect its functionality.

Symbol names must be unique within the entire hierarchy of the Graphic Toolbox.

To rename a symbol in the Graphic Toolbox

- 1 Open the Graphic Toolbox.
- 2 Select the symbol you want to rename.
- 3 On the **Edit** menu, click **Rename**. The symbol name is in edit mode.
- 4 Type a new unique name for the symbol and press **Enter**.

Copying Symbols

You can create copies of symbols in the Graphic Toolbox. The copies are suffixed with "_Copy1", "_Copy2", and so on.

To create a copy of a symbol

- 1 Select the symbol you want to copy.
- 2 On the **Edit** menu, click **Duplicate**. A copy of the symbol is created.
- 3 If needed, type a new name for the symbol.

Renaming Graphic Toolsets

You can rename a Graphic Toolset at any time. Renaming a Graphic Toolset does not affect the functionality of any symbols it contains.

To rename a Graphic Toolset in the Graphic Toolbox

- 1 Open the Graphic Toolbox.
- 2 Select the Graphic Toolset you want to rename.
- 3 On the **Edit** menu, click **Rename**. The Graphic Toolset name is in edit mode.
- 4 Type a new unique name for the Graphic Toolset and press **Enter**.

Deleting Graphic Toolsets

You can delete a Graphic Toolset in the Graphic Toolbox at any time. You can only delete Graphic Toolsets that do not contain any symbols. Move the symbols to another Graphic Toolset or delete them before deleting the Graphic Toolset.

To delete a Graphic Toolset in the Graphic Toolbox

- 1 Open the Graphic Toolbox.
- 2 Select the Graphic Toolset you want to delete.
- 3 On the **Edit** menu, click **Delete**. When a message appears, click **Yes**.

Moving Graphic Toolsets

You can move a Graphic Toolset within the Graphic Toolset hierarchy of the Graphic Toolbox. If you move a Graphic Toolsets, all symbols and Graphic Toolsets it contains are also moved.

To move a Graphic Toolset in the Graphic Toolbox

- 1 Open the Graphic Toolbox.
- 2 Locate the Graphic Toolset you want to move.
- 3 Do either of the following:
 - Drag it to the Graphic Toolset you want to place it in. The Graphic Toolset is moved to the new Graphic Toolset.
 - To move the Graphic Toolset to the top level in the Graphic Toolset hierarchy, drag the Graphic Toolset to the Galaxy name icon.

Customizing Graphic Toolsets

You can hide or show Graphic Toolsets after creating them. You can do this for selected Graphic Toolsets or for all Graphic Toolsets.

To customize Graphic Toolsets

- 1 Open the Graphic Toolbox.
- 2 Right-click the Galaxy icon in the Graphic Toolbox, and then click **Customize Toolsets**. The **Customize Toolsets** dialog box appears.
- 3 Do one or more of the following:
 - To hide Graphic Toolsets, clear the check box next to the Toolsets you want to hide.
 - To show Graphic Toolsets, check the check box next to the Toolsets you want to show.
 - To select or clear all Toolsets, click **Check All** or **Uncheck All**.
- 4 Click **Close**. The selected Graphic Toolsets are shown or hidden, depending on your settings.

Importing and Exporting Symbols as ArcestrA Object Files

You can import and export symbols as .aaPKG ArcestrA AutomationObject files.

Importing Symbols

You can import symbols and graphic toolsets from a symbol .aaPKG file.

When you import templates or instances that contain symbols, the symbols are imported with the template or instance. When you import all AutomationObjects, the contained symbols and the symbols in the graphic toolsets are also imported.

To import symbols from a symbol .aaPKG file

- 1 On the **Galaxy** menu, point to **Import**, and then click **Object(s)**. The **Import AutomationObject(s)** dialog box appears.
- 2 Select one or more symbol .aaPKG files you want to import and click **Open**. The **Import Preferences** dialog box appears.
- 3 Select the appropriate options for the import and click **OK**. The symbols and graphic toolsets are imported.
- 4 Click **Close**.

Exporting Symbols

You can export one or more symbols to a symbol .aaPKG file.

When you export object templates or instances that contain symbols, the symbols are exported with the template or instance.

When you export all AutomationObjects, the contained symbols and the symbols in the graphic toolsets are also exported.

When you export an AutomationObject that contains symbols, if these symbols contain embedded ArcestrA Symbols from the Graphic Toolbox, they are exported along with the symbols associated with the AutomationObject.

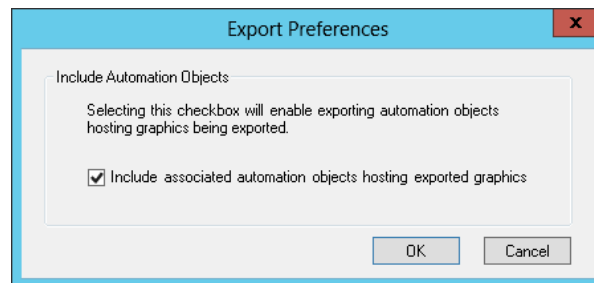
When you export an ArcestrA symbol containing embedded symbols associated with an object, you have the choice of including the object in the .aaPKG file.

If the symbols contain other AutomationObjects, the symbols, including any embedded symbols from the Graphic Toolbox contained in them, and their parent AutomationObject are not exported with the AutomationObject. They remain as references. Upon import, the system tries to reestablish the connection with those exact references in the target Galaxy.

To export symbols to a symbol .aaPKG file

- 1 In the Graphic Toolbox, select the symbols that you want to export.
- 2 On the **Galaxy** menu, point to **Export**, and then click **Object(s)**.

If you are exporting a symbol containing embedded symbols associated with an object, you see the **Export Preferences** dialog box with an option to include the associated automation objects hosting the graphics to be exported.



- 3 Select or clear the checkbox based on whether you want to export the automation objects or not and click **OK**.
- 4 The **Export Automation Object(s)** dialog box appears.
- 5 Browse to the save location and type a name for the symbol .aaPKG file.
- 6 Click **Save**. The symbols and their toolset folders are exported.
- 7 Click **Close**.

Programmatically Importing and Exporting ArcestrA Symbols

You can use the ArcestrA GraphicAccess application programming interface (API) to programmatically export a symbol from the ArcestrA Graphic Toolbox to an XML file. You can use the same API to import a graphic from an XML file to create an ArcestrA graphic in another galaxy or overwrite an existing graphic. You can also import a graphic XML file created by another application to an ArcestrA Galaxy.

The programmatic API exports or imports an extensive set of properties of an ArcestrA symbol. For standard ArcestrA symbols, an exported or imported symbol can contain the following:

- Custom properties
- Graphic elements
- Connector lines
- Graphic groups
- Symbol animations
- Element styles
- Named scripts
- Predefined scripts
- Overridden text strings
- Numeric format styles
- DataStatus elements
- Trend Pen
- Alarm Client
- Trend Client

In addition to the properties of a standard ArcestrA symbol, an exported or imported Symbol Wizard contains the following:

- Wizard Options
- Choice groups
- Choices
- Layers
- Rules

Implementing the GraphicAccess API

The GraphicAccess API is implemented by the ArcestrA.Visualization.GraphicAccess.dll file installed in the following folders based on 32-bit or 64-bit versions of Windows:

- 32-bit Windows: \Program Files\ArcestrA\Framework\Bin
- 64-bit Windows: \Program Files (x86)\ArcestrA\Framework\Bin

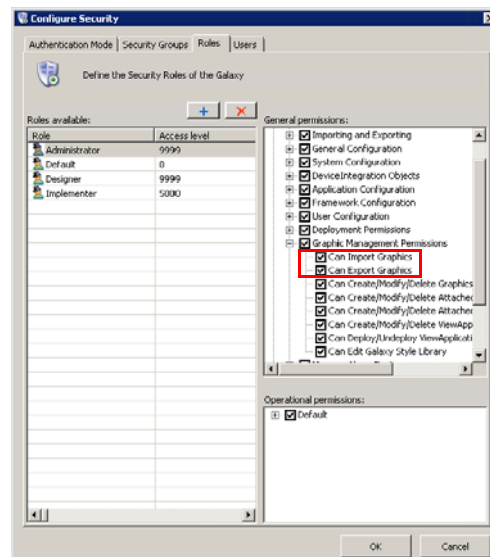
IGraphicAccess is an interface that is implemented by the GraphicAccess.dll component. IGraphicAccess is used to log in and connect to the ArcestrA Galaxy Repository.

The Galaxy name, graphic name, and the XML file path are passed as parameters to the methods of IGraphicAccess.

```
public interface IGraphicAccess
{
    /// <summary>
    /// Export an ArcestrA Galaxy graphic to Xml file
    /// </summary>
    /// <param name="galaxy">IGalaxy obtained from GRAccess</param>
    /// <param name="graphicName">The graphic name</param>
    /// <param name="xmlFilePath">The Xml file path</param>
    /// <returns>Result of the method</returns>
}
```

If the Galaxy has security enforced, you must first log in to the Galaxy Repository with proper credentials. Also, the **Can Export Graphics** and **Can Import Graphics** Galaxy role permissions must be set active before attempting an import or export operation.

You can set the **Can Export Graphics** and **Can Import Graphics** Galaxy role permissions from the ArcestrA IDE **Configure Security** dialog box. For more information about setting Galaxy permissions, see the *Application Server User Guide*.



GraphicAccess Interface Methods

The GraphicAccess interface includes separate graphic export and import methods.

ExportGraphicToXml Method

Exports an ArcestrA graphic to an XML file. The Galaxy name, graphic name, and the XML file path are passed as parameters of ExportGraphicToXml.

Syntax

```
ExportGraphicToXml(IGalaxy galaxy, string graphicName, string  
xmlFilePath);
```

Parameters

galaxy

Name of the Galaxy containing the graphic to export.

graphicName

Name of the graphic to export.

xmlFilePath

Directory folder to place the XML file containing the exported graphic.

ImportGraphicFromXml Method

Imports a graphic from an existing XML file. The Galaxy name, graphic name, the XML file path, and an overwrite flag are passed as parameters of ImportGraphicfromXml.

Syntax

```
ImportGraphicFromXml(IGalaxy galaxy, string graphicName, string  
xmlFilePath, bool bOverWrite);
```

Parameters

galaxy

Name of the Galaxy to import the graphic.

graphicName

Name of the graphic to import.

xmlFilePath

Directory folder location of the XML file.

bOverWrite

Boolean flag that indicates if an existing graphic can be overwritten by an imported graphic with the same name.

After an import or export operation is complete, the results are set to ICommandResult. A message appears and indicates if the operation succeeded or failed.

A succeeded message only means the import or export operation finished successfully. It does not indicate the quality of the exported XML file or the imported graphic. Check the SMC log file for any warning or error messages after each export or import operation.

Importing ArcestrA Symbols from XML Files

The `ImportGraphicFromXml` method imports an XML file and creates a graphic in the root folder of the ArcestrA Graphic Toolbox. A graphic container instance is created and loads the graphic definition from the provided XML file. The container then uses its binary serialization to save the binary definition to a blob. A new symbol is created with `GraphicAccess` API and the graphic blob is set to the Galaxy attribute "`_VisualElementDefinition`".

An XML schema file `aaGraphics.xsd` is located in the `..\ArcestrA\Framework\Bin` folder. The ArcestrA graphic library uses this schema file to validate the provided XML file before importing it and creating an ArcestrA graphic. If an invalid graphic name is specified or the XML file does not validate against the schema file, the import operation stops immediately without further processing.

Exporting ArcestrA Symbols to XML Files

During an export operation, the graphic definition is retrieved from Galaxy attribute "`_VisualElementDefinition`" of the graphic with the specified name. A graphic container instance is created for the exported graphic definition. XML serialization in the graphic container is used to create the graphic's XML file at the specified folder path.

Exporting and Importing Overridden Text Strings

The ArcestrA programmatic API supports overridden text strings in symbols. Typically, a string substitution is performed on a text string from the ArcestrA Symbol Editor before exporting the symbol with the programmatic API. The exported XML file shows the content of overridden and substituted text strings as elements and attributes of the `SubstitutedStrings` element.

When a symbol is imported using the programmatic API, the symbol shows the text string assigned to the `New` attribute of the XML file's `String` element. Users can substitute text strings before importing a symbol by editing the contents of the `String` element in the exported XML file.

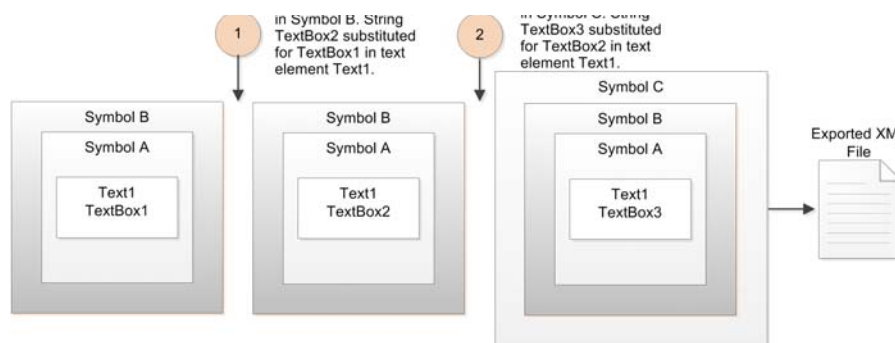
Exporting Overridden Text Strings

After exporting a symbol, the XML file's String element specifies the original string and the new substituted string with the Old and New attributes. The ElementID attribute specifies the specific text element to be overridden.

```
<SubstituteStrings>
  <String Old="TextBox1" New="TextBox2" ElementID="A.Text1"/>
</SubstituteStrings>
```

In this example, the TextBox1 text string was overridden by the new substituted text string TextBox2. The string substitution occurred in symbol B that substituted the text of the text element in symbol A.

Text strings can be overridden within symbols that are embedded within other symbols. Consider an example that shows symbol A embedded in symbol B, which is embedded in symbol C. Two string substitutions are made to a single text element.

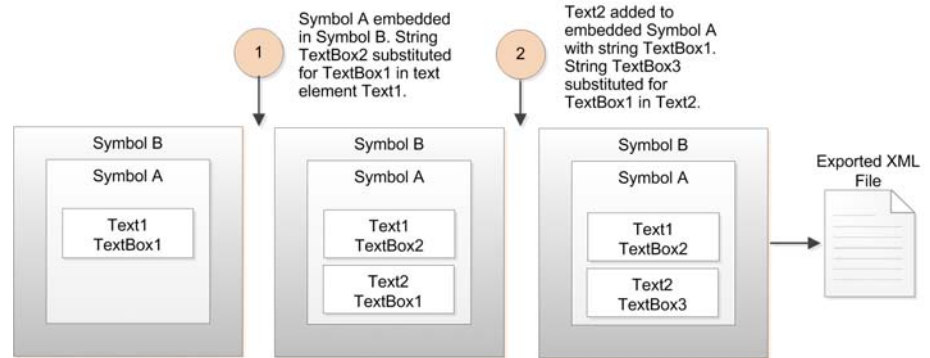


The export XML file's SubstituteStrings element shows the text string override information.

```
<SubstituteStrings>
  <String Old="TextBox2" New="TextBox3" ElementID="B.A.Text1"/>
</SubstituteStrings>
```

Notice the Old and New attributes show the text strings of the most recent string substitution before symbol C was exported. Also, the ElementID attribute indicates symbol A containing text element Text1 is embedded in symbol B in the form ElementID="B.A.Text1".

The programmatic API can export an embedded symbol containing multiple text elements with the same text string that is overridden by different string substitutions. Consider an example that shows symbol A embedded in symbol B. Two string substitutions are made to a single text string in two text elements of embedded symbol A.



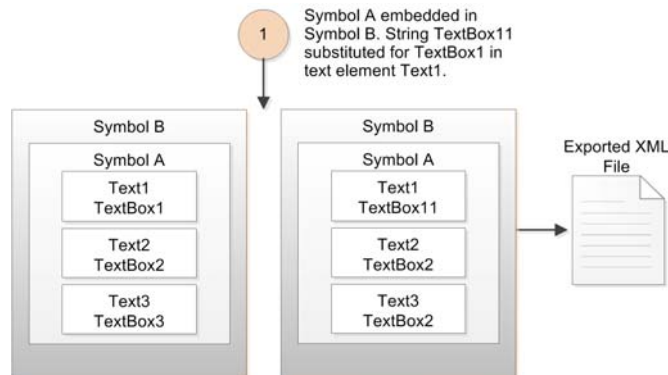
The export XML file's `SubstituteStrings` element shows the text string override information with a `String` element for each text override.

```
<SubstituteStrings>
  <String Old="TextBox1" New="TextBox2" ElementID="B.A.Text1"/>
  <String Old="TextBox1" New="TextBox3" ElementID="B.A.Text2"/>
</SubstituteStrings>
```

Importing Overridden Text Strings

An exported symbol's text strings can be overridden by editing the `SubstituteString` elements of the export XML file before importing the symbol.

The following illustration shows symbol A embedded in symbol B. Symbol A contains three text elements with text strings. The `TextBox1` text string of the `Text1` element was overridden to `Textbox11`. Then, symbol B was exported using the programmatic API.



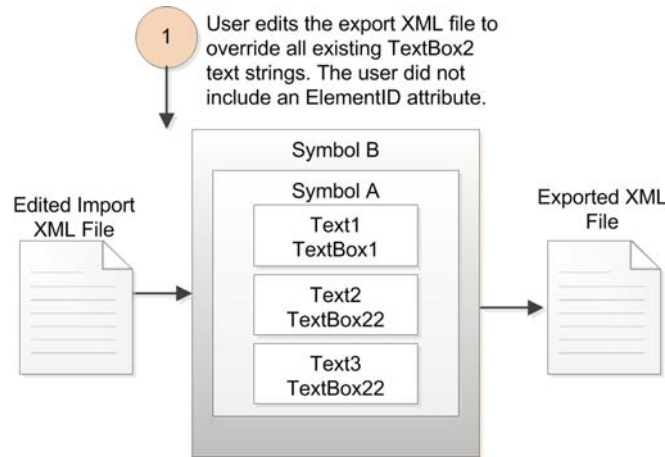
The expected `SubstituteStrings` element in the export XML file should be similar to the following:

```
<SubstituteStrings>
  <String Old="TextBox1" New="TextBox11" ElementID="A.Text1"/>
</SubstituteStrings>
```

The user edits the export XML file and adds an entry to override a text string without using an `ElementID` attribute to identify the text element.

```
<SubstituteStrings>
  <String Old="TextBox1" New="TextBox11" ElementID="A.Text1"/>
  <String Old="TextBox2" New="TextBox22">
</SubstituteStrings>
```

After importing the symbol, the text strings for the Text2 and Text3 elements are overridden to TextBox22. Without an ElementID to identify the text element, the string override replaces all text strings that match the text string specified by the Old attribute.



If the symbol was exported again using the programmatic API, the exported XML file shows that ElementID attributes have been added to each String element.

```
<SubstituteStrings>
  <String Old="TextBox1" New="TextBox11" ElementID="A.Text1"/>
  <String Old="TextBox2" New="TextBox22" ElementID="A.Text2"/>
  <String Old="TextBox2" New="TextBox22" ElementID="A.Text3"/>
</SubstituteStrings>
```

Associating All Galaxy Graphics with an InTouchViewApp

Use **Associate Galaxy Graphics** in the InTouchViewApp template to configure an "include all Galaxy graphics" option from the Graphic Toolbox, Template, and Instance objects in the InTouchViewApp. This option sets the InTouch View Application to include all the graphics that have been configured in the Galaxy whether they have been embedded in the application or not.

All Galaxy graphics will be included when the application is deployed or published. The "include all Galaxy graphics" option does not apply for export operations.

Note The term "graphic" includes any symbol or client control present in the Graphic Toolbox, and any symbols owned or inherited by templates and instances.

Access the **Associate Galaxy Graphics** dialog from the InTouchViewApp template and right-click context menu. For more information about associating all Galaxy graphics with an InTouchViewApp, see the *Application Server User's Guide*, Chapter 6, "Deploying and Running an Application".

Deleting a Symbol

You can delete a symbol that you no longer need. Deleting a symbol removes it completely from the Application Server. You can delete a symbol from the Graphic Toolbox or from an AutomationObject in the ArcestrA Symbol Editor.

- When you delete a symbol, you are shown where the symbol is used. This gives you an assessment of deleting the symbol before actually deleting it.
- You cannot delete symbols that someone else has open for editing or left checked out.
- If you delete a symbol from an AutomationObject, the symbol still appears to other users until you check in the AutomationObject.
- If you delete a symbol that is embedded in another symbol or in an InTouch window, it shows a Not Found message.

To delete a symbol

- 1 Do one of the following:
 - Open the Graphic Toolbox.
 - Open the AutomationObject with the symbols you want to delete. Click the **Graphics** tab.
- 2 Select the symbol you want to delete and click **Delete**. The **Delete** dialog box appears.
- 3 Review the places the symbol is being used, and then click **Yes**.

Creating Multiple Configurations of a Symbol

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

Symbol configurations are created using layers containing associated graphic elements, custom properties, and named scripts. Based on symbol properties and possible values of these properties, rules are applied that specify when a layer is part of a symbol configuration. For more information about Symbol Editor's window to assign symbol layers, graphic elements, rules, and properties, see "Using the Symbol Wizard Editor" on page 105.

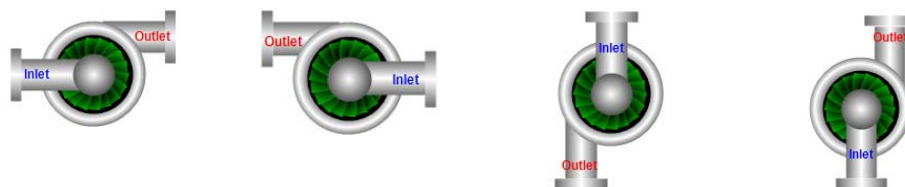
Understanding Visual and Functional Symbol Configurations

Standard ArchestrA symbols available from the IDE Graphic Toolbox show reasonably realistic views of process objects. These symbols can be modified with the Symbol Wizard Editor to incorporate multiple visual configurations in a symbol.

Situational Awareness Library symbols are designed using the Symbol Wizard Editor. However, they are protected symbols 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 symbol's design.

Visual Symbol Configurations

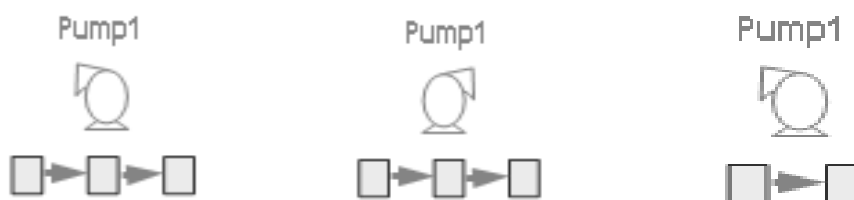
Using an example of a centrifugal pump with separate inlet and outlet pipes, there are four practical visual configurations that can be included in a single Symbol Wizard. The pump's blade housing is common and appears in all possible configurations. But, the pump's inlet and outlet pipes can be placed 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 symbol. The attributes associated with the Orientation property are left, right, top, and bottom.

Functional Symbol 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 three-stage or two stage pump in addition to a visual Orientation property to select left or right pump configurations.



Different Symbol Wizard Work Flows

There are two types of ArcestrA users who work with Symbol Wizards:

- Designers are ArcestrA users responsible for creating Symbol Wizards. After verifying that all configurations built for a symbol are correct, the Designer saves the Symbol Wizard to the Galaxy library just like standard ArcestrA symbols.

For more information about how a Designer works with the Symbol Wizard, see "Using the Symbol Wizard Editor" on page 105. To get more information about the procedures to create symbol configurations with the Symbol Wizard, see "Designing a Symbol Wizard" on page 480.

- Consumers embed Symbol Wizards with multiple configurations as part of creating managed InTouch applications. The Symbol Wizards's default configuration is selected when a symbol is embedded.

The Consumer can select a Symbol Wizard configuration based on the needs of the InTouch application. A Consumer selects the needed configuration by changing values or rules in the **Wizard Options** area of the Symbol Editor. For more information about how a Consumer works with the Symbol Wizard, see "Using Symbol Wizards in an Application" on page 489.

Embedded Symbols

You can embed symbols from the Graphic Toolbox, AutomationObject templates, and instances into other symbols. Embedding symbols enables you to rapidly develop more complex symbols with common components.

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

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

Appearance of Embedded Symbols

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

Changing Embedded Symbols

After you embed a symbol, you can change its size, orientation or transparency. You can add a limited set of animations to the symbol, 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 symbol from within the hosting symbol.
- Embed a symbol contained in an AutomationObject into a symbol that is contained in the Graphic Toolbox.
- Create circular references. A circular reference occurs when one symbol (Symbol A) has embedded within it another symbol (Symbol B) that has embedded within it a symbol that directly or indirect references back to the first symbol (Symbol A).

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

Configuring Security for Symbols

You can set IDE security permissions so that at design time, the user cannot:

- Import or export symbols.
- Create, modify, or delete symbols in the Graphic Toolbox.
- Create, modify, or delete symbols in any AutomationObject template.
- Create, modify, or delete symbols in any AutomationObject instance.
- Create, modify, or delete View Applications, such as InTouchView Applications.

- Deploy or undeploy View Applications, such as InTouchView Applications.
- Edit the configuration of the quality and status display.

To restrict a user, the user must be assigned to a role and the permissions must be assigned to that role, and security must be enabled. For more information on how to configure security, users, and roles, see the *Application Server User's Guide*.

If the user attempts to export a symbol without appropriate permissions, the message "User doesn't have permission to export graphics object" appears.

If the user attempts to import a symbol without appropriate permissions, the message "User doesn't have permission to import graphics object" appears.

To configure security for symbols

- 1 On the **Galaxy** menu, point to **Configure**, and then click **Security**. The **Configure Security** dialog box appears.
- 2 Click the **Roles** tab.
- 3 In the **Roles available** list, click the role you want to assign the permissions to.
- 4 In the **General Permissions** list, expand the **Graphic management permissions**.
- 5 If you want to restrict the user from:
 - Importing symbols, clear the **Can import graphics** check box.
 - Exporting symbols, clear the **Can export graphics** check box.
 - Creating, modifying, or deleting symbols within toolsets, clear the **Can Create/Modify/Delete graphics within toolsets** check box.
 - Creating, modifying, or deleting symbols attached to a template, clear the **Can Create/Modify/Delete attached object graphics in template** check box.
 - Creating, modifying, or deleting symbols attached to an instance, clear the **Can Create/Modify/Delete attached object graphics in instance** check box.
 - Creating, modifying, or deleting view applications, clear the **Can Create/Modify/Delete ViewApplications** check box.
 - Deploying and undeploying view applications, clear the **Can Deploy/Undeploy ViewApplications** check box.
 - Accessing the edit quality and status display configuration, clear the **Can Edit Quality and Status Indicator Configuration** check box.
- 6 Click **OK**.

Writing to Attributes Configured for Secured or Verified Writes

There are several ways to write to an Attribute configured for Secured Write or Verified Write security classification.

- Any assignment in a script that sets the value of the Attribute, such as
`A=B;`
where A references an Attribute that is configured for Secured Write or Verified write security classification.
- Any action on an animation graphic that alters the value of an Attribute that has Signed Write or Verified Write security configuration, such as a user input, a slider, an up/down button on a counter, and other such actions.
- A script that uses the SignedWrite() function.

For information specific to the SignedWrite() function, see "Working with the SignedWrite() Function for Secured and Verified Writes" on page 89.

Working with the SignedWrite() Function for Secured and Verified Writes

This section provides information about the SignedWrite() function, SignedWrite() run-time behavior, scripting tips, and in-depth script examples.

For SignedWrite() scripting information including script syntax, parameters, and basic script examples, see "SignedWrite()" in Chapter 2, "QuickScript .NET Functions," in the *Application Server Scripting Guide*.

You can write to an Automation Object attribute that is configured for Secured Write or Verified Write security classification by means of the ArcestrA Graphics SignedWrite() script function.

The SignedWrite() function can be used only in ArcestrA client scripts, not in Application Object scripts, and only on Attributes that have been configured for Secured Write or Verified Write. Attempting to use the function on an Attribute not so configured will result in an error message at run time.

SignedWrite() Run-time Behavior

At run time, the SignedWrite() function does the following:

- 1 Checks the target Attribute for Signed Write or Verified Write configuration.

If not so configured, the following error message appears:

Operation Failed. Attribute does not have the correct security classification.

- 2 Checks Galaxy security.

If the Galaxy is not secured, the following error message appears:

Operation Failed. Galaxy is not secured.

- 3 There are several ways to write to an Attribute configured with Secured Write or Verified Write security classification, it is possible to have multiple SignedWrite() and other Secured/Verified writes pending from the same script, or even from multiple scripts running side-by-side.

- 4 Determines which dialog is required—Secured Write or Verified Write—and pops up the appropriate dialog box.

If Smart Cards are enabled, the function displays different versions of the Secured Write and Verified write dialog boxes.

- 5 Lists the predefined comments, if any, from the configured Predefined Comments list. Up to 20 comments are supported.
- 6 Enables comment editing if the Comment_Is_Editable parameter is configured and comment enforcement is other than PredefinedOnly.

- 7 Acquires the user credentials and authenticates them.

If the user credentials are invalid, an error message appears. The function will attempt the write only if the credentials are valid.

- 8 Checks if comment enforcement is mandatory, and displays an error message if the comment is empty.
- 9 Performs the write if user credentials are valid and the comment entry satisfies the comment enforcement parameter.
- 10 Provides a return status.

- 11 Following a Secured or Verified Write a security Event is written to the event log, including the signee name, verifier name, if any, Type of write: "Secured Write" or "Verified Write", Comment, if any entered by user, Reason Description, if any provided, Field Attribute description, if any, or the Short Description of the Application Object, if no Field Attribute description exists.

Each call to SignedWrite() is distinct from any other. The success or failure of any individual write does not affect other attempted writes.

Entering user credentials for SignedWrite() is distinct from logging on to the client application. The user can modify attributes configured with Secured or Verified Write even if another user is logged on, without affecting the session of the logged-on user.

SignedWrite() Script Execution Sequence at Run Time

The SignedWrite() function goes into a queue and the script continues executing. The function is queued for operator entry. The script may complete prior to the operator completing the Secured or Verified Write operation.

By contrast, the SignedAlarmAck() script function executes completely synchronously, and waits for user input before proceeding to the next line in the script.

SignedWrite() Scripting Tips

Using Bound References in SignedWrite()

If the Attribute parameter string evaluates to the name of a Custom Property and that Custom Property is a bound reference to an Attribute, the SignedWrite() function will write to that indicated Attribute. The Attribute must have the security classification of Secured Write or Verified Write.

The SignedWrite() function supports Custom Properties that are nested bound references. That is, if the string evaluates to the name of a Custom Property and that Custom Property is a bound reference to another Custom Property which itself is a bound reference, the SignedWrite() function will follow through the chain of bound references until it finds an item that is a value. If that item is an Attribute that has the security classification of Secured Write or Verified Write, the SignedWrite() function will write to that item.

Using SignedWrite() in WhileTrue, WhileFalse, or Periodic Type Scripts

Using the SignedWrite() function with WhileTrue, WhileFalse, or Periodic type scripts can repeatedly execute the script, causing another secured write dialog box to pop up with each trigger. We do not recommend using the SignedWrite() function with WhileTrue, WhileFalse, or Periodic types.

Using SignedWrite() with OnShow and OnHide Scripts

We do not recommend using the SignedWrite() 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.

Examples of Using the Attribute Parameter in the SignedWrite() Function

Working from the overall syntax of the SignedWrite() function, the script examples in the following table illustrate a number of approaches to using the *Attribute* parameter in the SignedWrite() function.

The following String, Boolean, and Integer user defined attribute conditions apply to the script examples:

- User Defined object UD1_001
- String attribute UD1_001.udString1 with the value "WW". Security classification is set to Secured Write.
- Boolean attribute UD1_001.secBool1 with the value false. Security classification is set to Secured Write.
- Integer attribute UD1_001.secInt1 with the value 24. Security classification is set to Secured Write.

• User Defined object UD2_002
String attribute UD2_002.udString2 with a value "UD1_001.udString1" The following Custom Property conditions apply to the script examples:

- String Custom Property CP1 with a reference to UD1_001.udString1
- String Custom Property CP2 with a value "UD1_001.udString1"
- String Custom Property CP3 with a value "UD1_001"
- Boolean Custom Property CP4 with a reference to UD1_001.secBool1
- Integer Custom Property CP5 with a reference to UD1_001.secInt1
- String Custom Property CP6 with a reference to an attribute on Owing Object UD1_001 as me.udString1

| Script Example | Function and Result |
|--|--|
| <pre>SignedWrite("CP1", "Invensys", "using redirect", true, 0, null);</pre> | <p>Uses the CP1 reference UD1_001.udString1 and pokes to it the value "Invensys".</p> <p>Result: The value in UD1_001.udString1 will change from "WW" to "Invensys".</p> |
| <pre>SignedWrite(CP2, "Invensys", "using string value", true, 0, null);</pre> | <p>Resolves CP2 string value "UD1_001.udString1" to a reference and pokes to it the value "Invensys".</p> <p>Result: The value in UD1_001.udString1 will change from "WW" to "Invensys".</p> |
| <pre>SignedWrite(CP3+".udString1", "Invensys", "using string expression", true, 0, null);</pre> | <p>Resolves the string "UD1_001.udString1" to a reference and pokes to it the value "Invensys".</p> <p>Result: The value in UD1_001.udString1 will change from "WW" to "Invensys".</p> |
| <pre>SignedWrite("UD1_001.udString1", "Invensys", "using constant string", true, 0, null);</pre> | <p>Resolves the string "UD1_001.udString1" to a reference and pokes to it the value "Invensys".</p> <p>Result: The value in UD1_001.udString1 will change from "WW" to "Invensys".</p> |
| <pre>SignedWrite(UD2_002.udString2, "Invensys", "using attribute containing string", true, 0, null);</pre> | <p>Resolves the UD2_002.udString2 string value "UD1_001.udString1" to a reference and pokes to it the value "Invensys".</p> <p>Result: The value in UD1_001.udString1 will change from "WW" to "Invensys".</p> |
| <pre>SignedWrite("CP" + "1", "Invensys", "using redirect from string expression", true, 0, null);</pre> | <p>Resolves the expression to "CP1" to use the CP1 reference UD1_001.udString1 and pokes to it the value "Invensys".</p> <p>Result: The value in UD1_001.udString1 will change from "WW" to "Invensys".</p> |
| <pre>SignedWrite("CP4", true, "using redirect", true, 0, null);</pre> | <p>Uses the CP4 reference UD1_001.secBool1 and pokes to it the value true.</p> <p>Result: The value in UD1_001.secBool1 will change from false to true.</p> |

| Script Example | Function and Result |
|--|---|
| <pre>SignedWrite("CP5", 37, "using redirect", true, 0, null);</pre> | <p>Uses the CP5 reference UD1_001.secInt1 and pokes to it the value 37.</p> <p>Result: The value in UD1_001.secInt1 will change from 24 to 37.</p> |
| <pre>SignedWrite("CP6", "Invensys", "using redirect using relative reference", true, 0, null);</pre> | <p>Uses the CP6 reference me.udString1 and resolves it to UD1_001.udString1 and pokes to it the value "Invensys".</p> <p>Result: That the value in UD1_001.udString1 will change from "WW" to "Invensys".</p> |

Secured and Verified Write Applied Examples

You can create a dashboard application to automate routine use of Secured and Verified Write by means of the SignedWrite() function.

To configure the SignedWrite() script function

- 1 Open the ArcestrA IDE.
- 2 Create a symbol and associate it with an attribute configured with Secured Write or Verified Write. For more information on associating attributes with symbols, see *Application Server User's Guide*, "Creating and Working with UDAs" topic.
- 3 Add the SignedWrite script function to the symbol. The following editor detail shows the buttons configured with scripts in the applied example:
- 4 Configure the scripted functionality you require. Scripts for the buttons shown in the example are as follows:

- c** Hard-coded DataUDO.SecUDA: The following example sets the value of 23 to DataUDO.SecUDA. The user optionally can enter a comment, but no pre-defined comment list is available.

```
DataUDO.RetStatus=SignedWrite("DataUDO.SecUDA", 23, "Set
the Value", True, 0, null);
```

- d** Attribute Pointer has DataUDO.SecUDA: The source to be written to is passed as a parameter to the function. Attribute_Pointer is a Custom Property whose value is set to DataUDO.SecUDA.

The following example sets the value of 23 to DataUDO.SecUDA. The user optionally can enter a comment, but no pre-defined comment list is available.

```
DataUDO.RetStatus=SignedWrite(Attribute_Pointer, 23,
"Set the Value", True, 0, null);
```

- e** **Attribute Pointer and Pre-Defined List:** The pre-defined comment list is an array. This example extends the functionality of example b to force the user to enter a comment (Comment_Enforcement parameter set to 1) and also presents a pre-defined set of comments linked to the DataUDO.PreDefComments[] array.

The following example will set the value of 23 to DataUDO.SecUDA. The user must enter a comment and may use one from the pre-defined comment list.

```
DataUDO.RetStatus=SignedWrite(Attribute_Pointer, 23,
"Set the Value", True, 1, DataUDO.PreDefComments[ ]);
```

- f** **Variable Array:** The pre-defined list is a pointer to an array. This example extends the functionality of example c to force the user to enter a comment (Comment_Enforcement parameter set to 1) and also presents a predefined set of comments linked to DataUDO.PreDefComments[] array.

The value of Custom Property CP1 is "DataUDO.PreDefComments[]".

The following example will set the value of 23 to DataUDO.SecUDA. The user must enter a comment and may use one from the pre-defined comment list.

```
dim xInd as Indirect;
xInd.BindTo(CP1);

DataUDO.RetStatus=SignedWrite(Attribute_Pointer, 23,
"Set the Value", True, 1, xInd);
```

- g** **All Parameters Variable:** The predefined list array is built into the script. All parameters are passed as variables.

The following example will set the value of 23 to DataUDO.SecUDA. The user must enter a comment and may use one from the pre-defined comments list.

```
dim MyList[5] as string;
MyList[1] = "Batch Accepted";
MyList[2] = "Batch Rejected";
MyList[3] = "Batch on Hold";
MyList[4] = "Batch Resumed";
MyList[5] = DataUDO.PreDefComments[4];

DataUDO.RetStatus=SignedWrite(Attribute_Pointer,
SignedWrite_Value_Ptr, SignedWrite_Reason,
Enable_Edit_Comment, Comment_Options, MyList[ ]);
```

Viewing a Symbol in Read-Only Mode

You can view a symbol 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 symbol in read-only mode, you have access to all functions in the ArcestrA Symbol Editor that do not change the symbol.

To view a symbol in read-only mode

- 1 In the Graphic Toolbox, select the symbol that you want to view in read-only mode.
- 2 On the **Galaxy** menu, click **Open Read-Only**. The selected symbol opens in the ArcestrA Symbol Editor.

Chapter 4

Using the ArcestrA Symbol Editor

You can edit ArcestrA Symbols using the ArcestrA Symbol Editor. Depending on where the ArcestrA Symbol is contained, you can start the ArcestrA Symbol Editor from:

- The Graphic Toolbox.
- The **Graphics** tab of an AutomationObject template.
- The **Graphics** tab of an AutomationObject instance.
- An embedded ArcestrA Symbol in an InTouch window.

You can:

- Show and hide ArcestrA Symbol 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.

Showing, Hiding, and Adjusting Panels

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 symbol. 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

You can 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

You can 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 pressing the mouse button.
- 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

You can 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 press **Enter**.

Using the Pan and Zoom Window to Change the Zoom

You can 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 98.

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 Symbol 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 symbol 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 Symbol 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 176.
- 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 symbol 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 176.
- 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 symbol displays anchor icons, if anchors were created in the symbol.

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 symbol 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 44."

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

Designers and Consumers work with the Symbol Wizard Editor to create symbols with multiple configurations called Symbol Wizards. Designers create Symbol Wizards with the Symbol Wizard Editor. Consumers embed Symbol Wizards and use the Symbol Wizard Editor to select the configuration needed for an application.

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

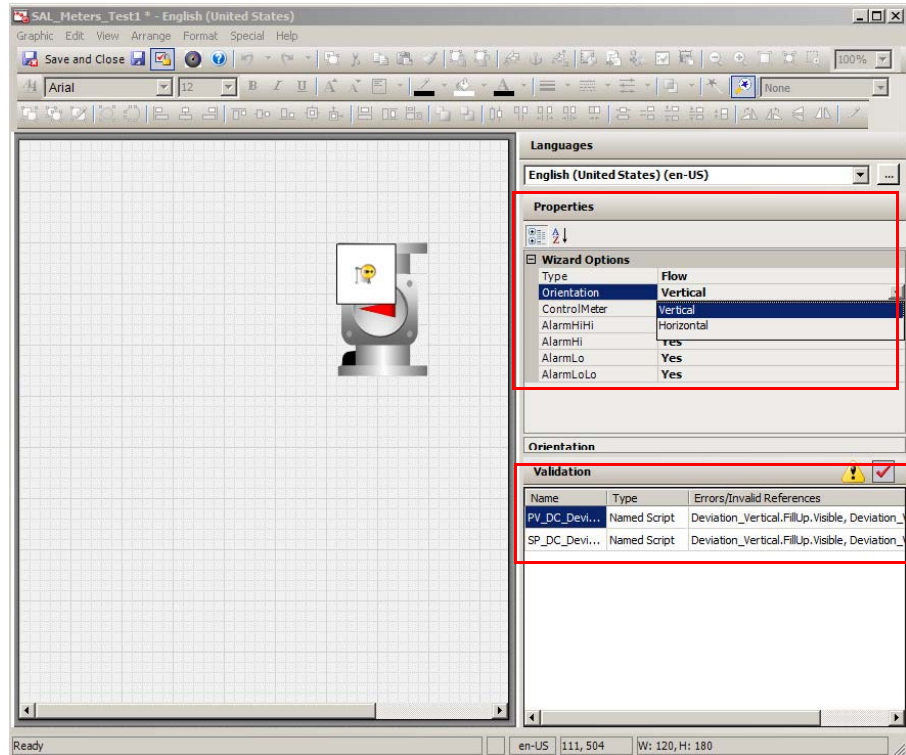
The Symbol Editor window updates to show tabbed Symbol Wizard panes at the left of the window. The top pane shows the graphic elements, named scripts, and custom properties of the symbol in separate views.

The bottom pane shows tabbed **Options** and **Layers** panes. The **Options** pane shows a hierarchical list of Choice Groups, Choices, and Options that define symbol properties and the possible values associated with each property. The **Layers** pane includes a list of defined symbol layers. Beneath each layer, separate folders contain the symbol's graphic elements, custom properties, and named scripts associated with each layer. Designers 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 Symbol 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 symbol with the Symbol Wizard Editor, Designers 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.

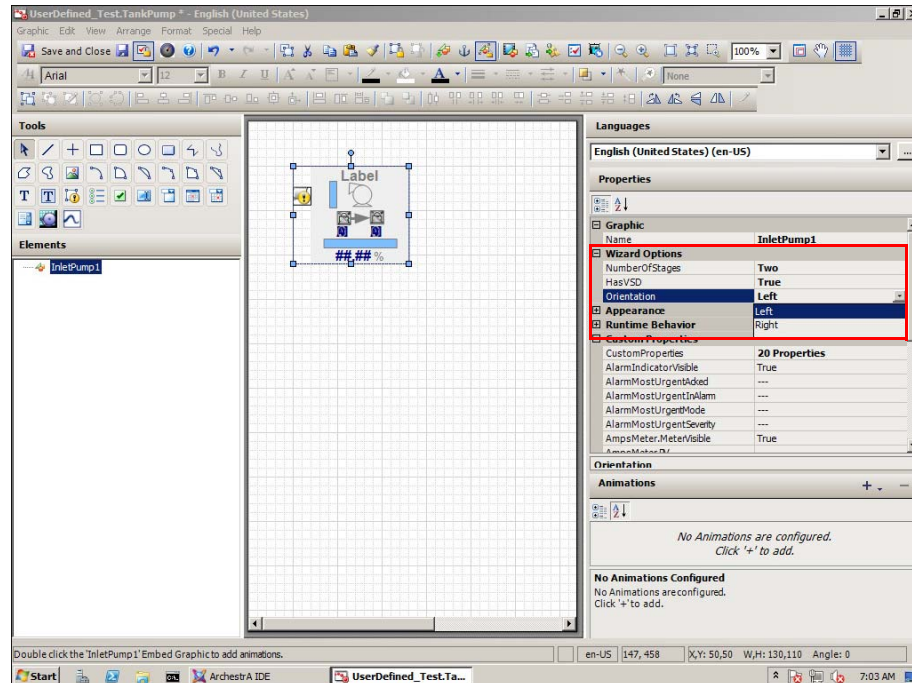


After opening Symbol Wizard Preview, the **Properties** pane shows **Wizard Options**, which includes drop-down menus to select options to show the different configurations created for the symbol. As options are selected, the symbol updates to show the selected configuration.

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

After verifying that all symbol configurations are correct, Designers save the symbol into the Graphic Toolbox. For more information about the Symbol Wizard tasks completed by a Designer, see "Designing a Symbol Wizard" on page 480.

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



The **Wizard Options** pane shows a set of drop-down lists with configuration options. Consumers select options from the drop-down lists to change the symbol's configuration to meet the needs of an application. Finally, the Consumer edits and updates the custom properties and named scripts that are associated with the multi-configuration symbol. For more information about the Symbol Wizard tasks completed by a Consumer, see "Using Symbol Wizards in an Application" on page 489.

Chapter 5

Working with 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 "Setting Symbol and Element-Specific Properties" on page 209.

About Graphic Elements

Graphic elements are basic shapes and controls you can use to create a symbol 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 symbol.
- 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 114.

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 Right-click when you are done.
- 6 You can change the shape of these elements anytime by editing their control points. For more information, see "Editing Control Points" on page 218.

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 218.

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 218.

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 217.

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 press 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 217.

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 press Enter.

Drawing Status Elements

You can 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 Windows Controls

You can draw Windows controls on the canvas to add additional functionality to your symbol. Each of the Windows 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 Windows control icon in the Tools panel.
- 2 Click the canvas where you want to place the Windows 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/or properties in the Properties Editor.

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 |
|--------------------------|---|
| Graphic | Element name or other describing identifiers |
| Wizard Options | Set of drop-down lists to select different symbol property attributes and options for Symbol Wizard configurations. |
| Appearance | Element location, size, orientation, offset, transparency and locked status |
| Fill Style | Any parameters related to the fill appearance of the element |
| Line Style | Any parameters related to the line appearance of the element |
| Text Style | Any parameters related to the text appearance of the element |
| Runtime Behavior | Element visibility, tab order and any other element behavior at run time |
| Custom Properties | Additional user-defined properties you can associate with any element |

For more information on the individual properties for each element, see "Alphabetical List of Properties" on page 505.

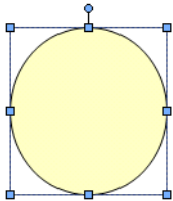
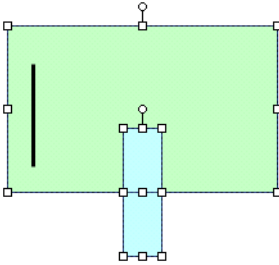
Selecting Elements

You can select one or more elements by:

- Clicking on them with the mouse.
- Dragging the lasso around them with the mouse.
- Selecting them with a menu option or with a shortcut key.
- Selecting them in the Elements List.

When you select an element, it appears with handles that give you control over its size and orientation.

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

| Selected Element | Description |
|--|---|
| Primary Element | Appears with color-filled handles. |
|  | Behaves as an active selected element. Is the point of reference for all operations, such as aligning or spacing multiple selected elements. |
| Secondary Elements | Appear with white handles. |
|  | Behave as inactive selected elements. Follow the edits made to the primary element. |

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 certain elements on the canvas, you can edit them by selecting them and clicking on them again. This is called inline editing. The following elements can be edited in this way:

| 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. |
| 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. The element handles appear.
- 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. Use these 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 218.

 - For arcs, pies, and chords, the handles for the start angle and sweep angle appear. Use these 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.

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 ArchemstrA Symbol 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. For more information about using the grid, see "Configuring Designer Preferences" on page 103.

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 press 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 spacial 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 115.

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 **Send 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 134.

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 Press 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 132.

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

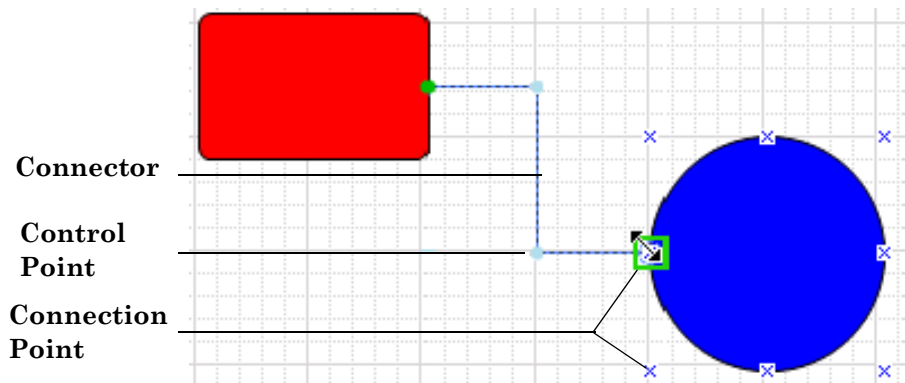
- 1 Select one or more elements on the canvas.
- 2 In the Properties Editor, do one of the following:
 - Type the absolute coordinates in the x, y format for the point of origin.
 - Type the relative coordinates in the x, y format for the point of origin.

- 3 Press 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.

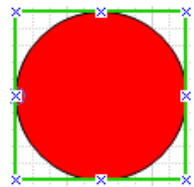
Adding Connectors Between Graphic Elements

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 symbols 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 symbol 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 symbol.



You can also add custom connection points to graphic elements or embedded symbols if you want to place a connector at a different position than the 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: Angled connector lines do not maintain their horizontal and vertical orientation with 90 degree angles when placed in a symbol 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 path. For more information about adding control points to a connector, see "Changing the Shape of a Connector" on page 145.

Connectors can be exported or imported with the ArcestrA GraphicAccess application programming interface (API). You can programmatically export a symbol containing connectors from the ArcestrA Graphic Toolbox to an XML file. You can use the same API to import a graphic containing connectors from an XML file to create an ArcestrA graphic in another galaxy or overwrite an existing graphic.

Drawing a Connector

You 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 Symbol Editor's canvas takes you out of connector drawing mode.

To draw a connector

- 1 Open a symbol in Symbol Editor that you want to add a connector.
- 2 Click once on the **Connector** icon within the **Tools** pane to draw a single connector.

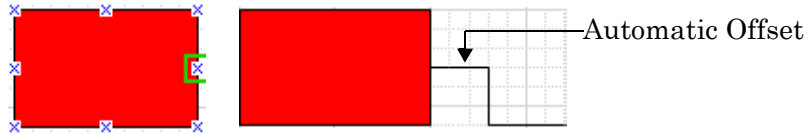
If you want to draw multiple connectors, double click on the **Connector** icon.

- 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

You 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 symbol 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 Symbol Editor's canvas takes you out of connection point addition mode.

To add connection points

- 1 Open a symbol 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 symbol.

- 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 symbol.

- 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.

Using Custom Connection Points with Embedded and Grouped Symbols

You can attach connectors to custom connection points in complex symbols, including embedded and grouped symbols, symbols embedded within other symbols, and complex Symbol Wizards.

All connector and connection point functionality remains available in embedded and grouped symbols.

- Create custom connection points on a symbol, draw connectors to another symbol, and save the symbols and their connection points.
- Group or ungroup connected symbols and save the symbol group(s) and their connection points and connectors.
- Connect a symbol within a group to custom connection points on another symbol within the group.
- Connect a symbol within a group to custom connection points on symbols in a different symbol group.
- Move grouped or individual symbols with their connectors dynamically intact.
- Undo symbol group conversion, and the custom connection points and connectors remain intact.
- Change symbol grouping, and the custom connection points and connectors remain intact.

Custom connection point functionality with embedded and grouped symbols also applies to Symbol Wizards. For example, you can attach a connector to a Symbol Wizard containing custom connection points. When you drag one end of a connector over the Symbol Wizard instance, it will highlight the custom connection points and you can attach a connector to one of them.

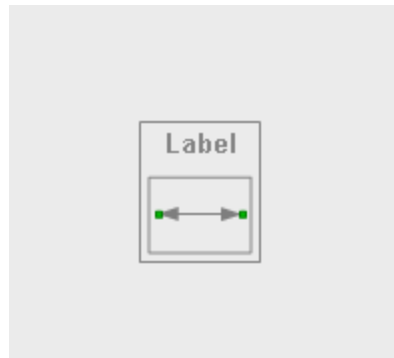
A connector will be disconnected only if you delete the custom connection point from the source symbol or if you change the containment of the custom connection point in the source symbol through operations such as group, ungroup, and convert symbol into group.

Using Connection Points with Embedded Situational Awareness Library Symbols

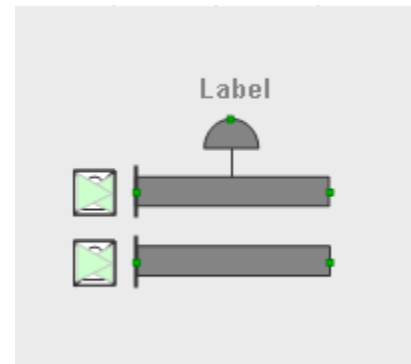
Situational Awareness Library symbols are protected to prevent them from being modified.

A set of 18 Situational Awareness Library symbols have connection points by default, based on the most common locations of equipment connections. These are:

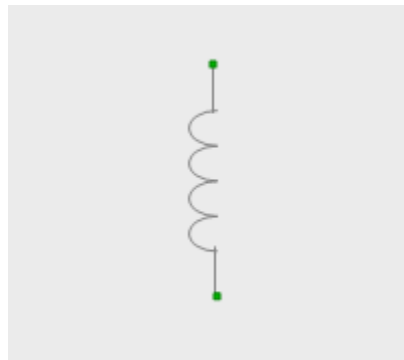
SA_DirectionArrow



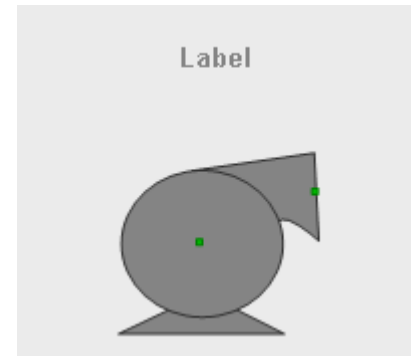
SA_ParallelControlValve



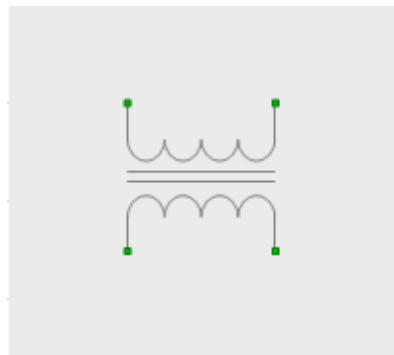
SA_ElectricalReactor



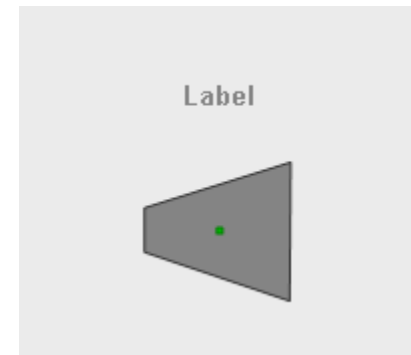
SA_Pump_Blower_RotaryValve



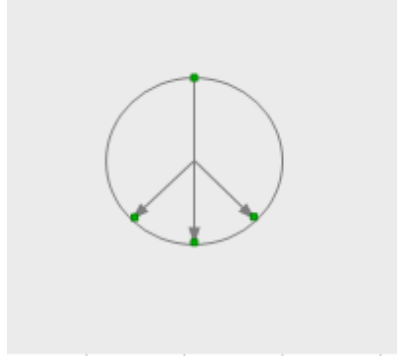
SA_ElectricalTransformer



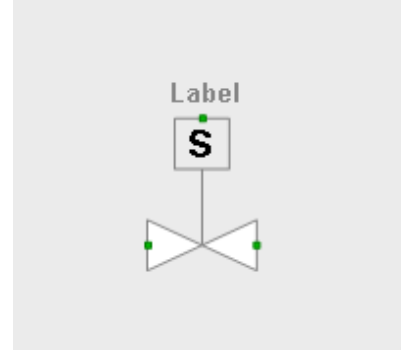
SA_RotatingEquipment



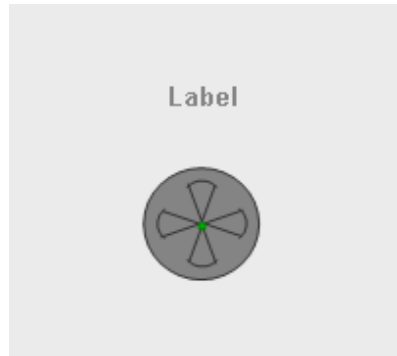
SA_HandSwitchSelector



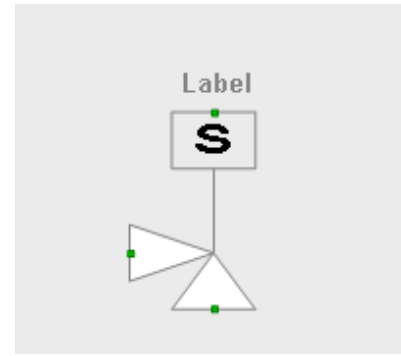
SA_Valve_2Way



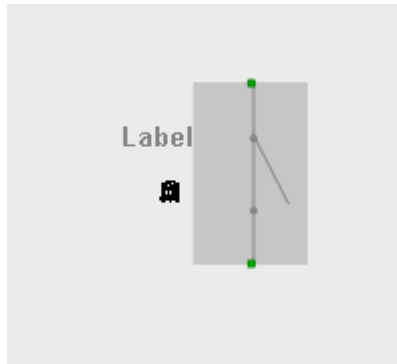
SA_HeatExchanger_Fan



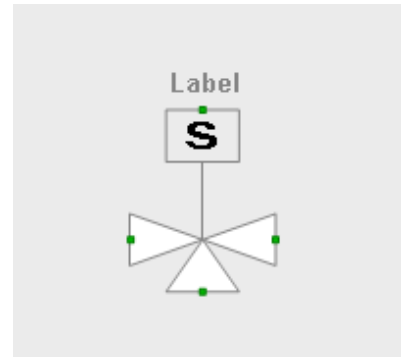
SA_Valve_2WayAngle



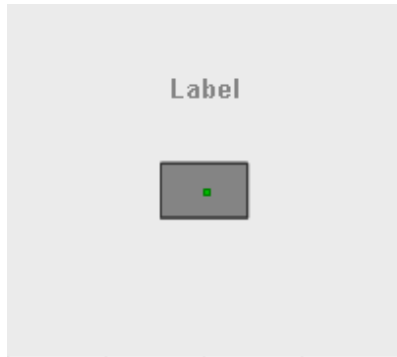
SA_HVLV_Switch



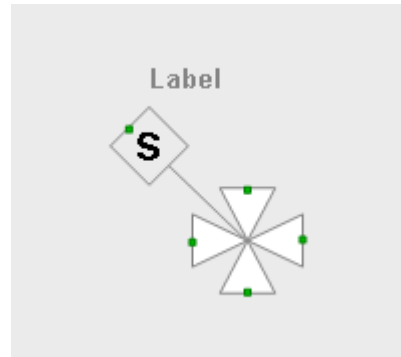
SA_Valve_3Way

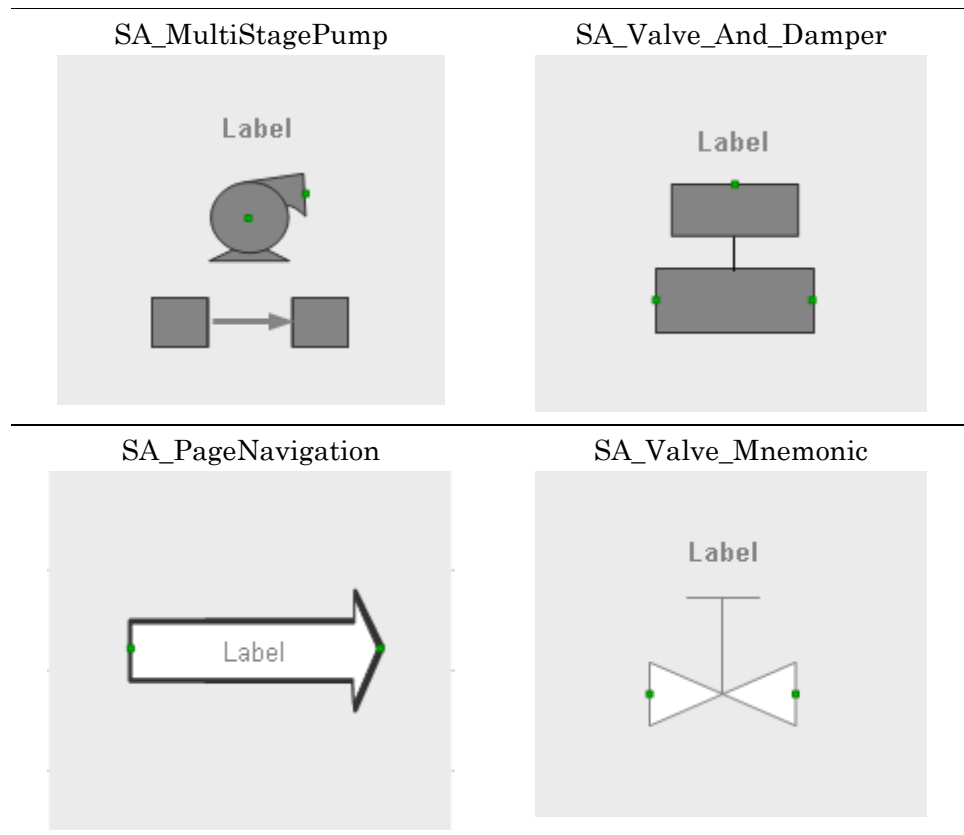


SA_MiscellaneousEquipment



SA_Valve_4Way





A Symbol Wizard option, **ConnectionPoint**, enables you to show or hide the connection points in the ArchestrA Graphic Editor during configuration. The default value of this Wizard Option is True.

The ConnectionPoint Wizard Option is tied to other Wizard Options such as Orientation and Type.

Some connection points will be tied to additional Wizard options, and will be available only when that specific indicator is enabled.

Tips for using connection points with embedded Situational Awareness Library symbols:

- If undesired connectors appear on top of an embedded symbol, use the **Order** option from the **Arrange** menu and select **BringToFront** for the symbol or **SendToBack** for the specific connector.
- To eliminate undesired connection elbows, use connector type Straight.

Changing Connector Properties

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

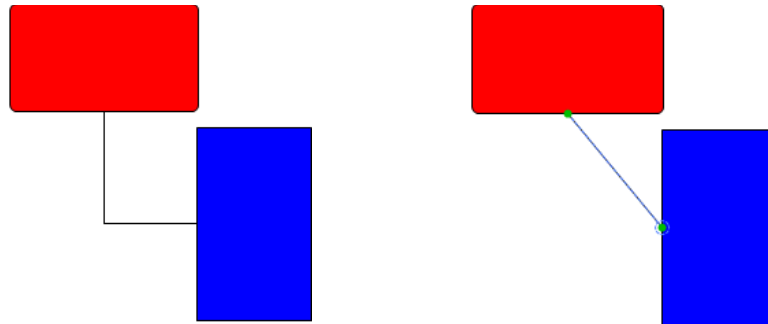
During run time, you can use animation to change property values that affect the appearance or behavior of a connector. For Angled or Straight connectors, you can use Line Style or Element Style animation. A connection point does not support any type of animation.

| Property | Description |
|------------------------------------|---|
| Appearance Properties | |
| ConnectionType | Type of connector (Angled or Straight). Angled is the default. |
| 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. |
| 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 Symbol 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 Symbol Editor's canvas. |
| Line Style Properties | |
| 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. |
| Runtime Behavior Properties | |
| Enabled | Connector animation is enabled or disabled during run time. Enabled is the default. |
| Visible | Connector is visible or hidden during run time. Visible is the default. |

Changing 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 Symbol 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 Symbol 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.

Changing 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.

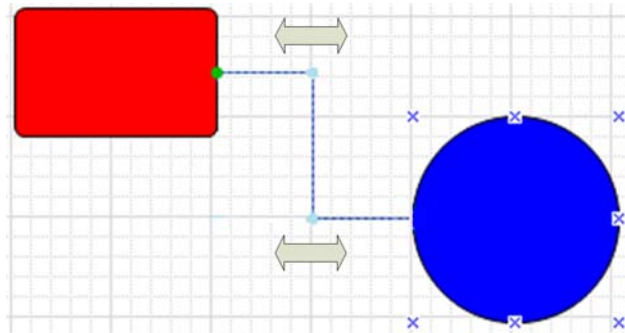
Changing the Shape of a Connector

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 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.

If you want to be able to move the two horizontal line segments of the connector vertically in the following example, you must add control points.



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.

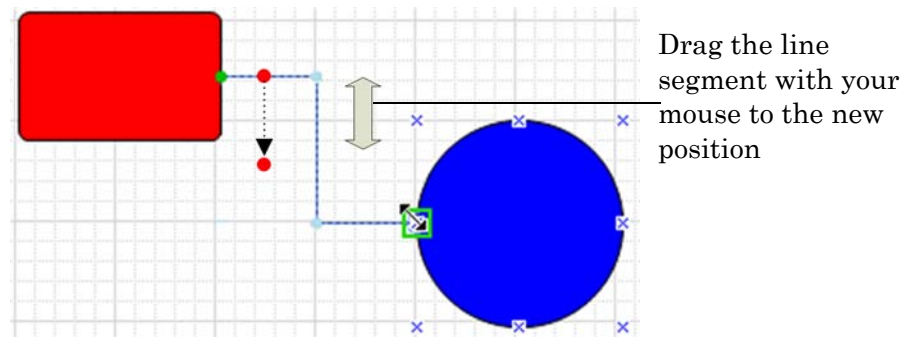
- 2 Click a control point and keep your mouse key pressed.

The mouse cursor changes to a double arrow to indicate the control point can be moved.

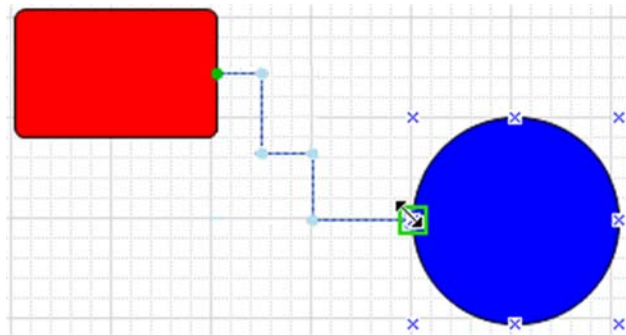
- 3 Move the control point to change the shape of the connector and release the mouse key.

Adding a Control Point

You can add control points to be able to change a connector path that is not possible using only the default set of control points. The following figure shows a control point added to the first horizontal line segment of a connector.



In this example, adding a control point enables the line segment between the two adjacent control points to be moved vertically. You move a line segment between two control points by pressing your mouse key and dragging the line segment to the new position.



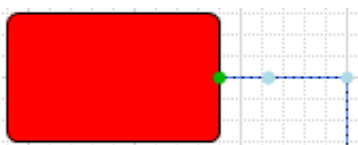
To add a control point to a connector

- 1 Click on a connector to select it.
- 2 With your Shift key pressed, place your cursor over the connector at the position where you want to add a control point.

The appearance of the cursor changes to a pen tip with a plus sign to indicate that a control point can be added.

- 3 Left click with your mouse to add a control point.

A small blue circle on the connector indicates the position of the control point you added.

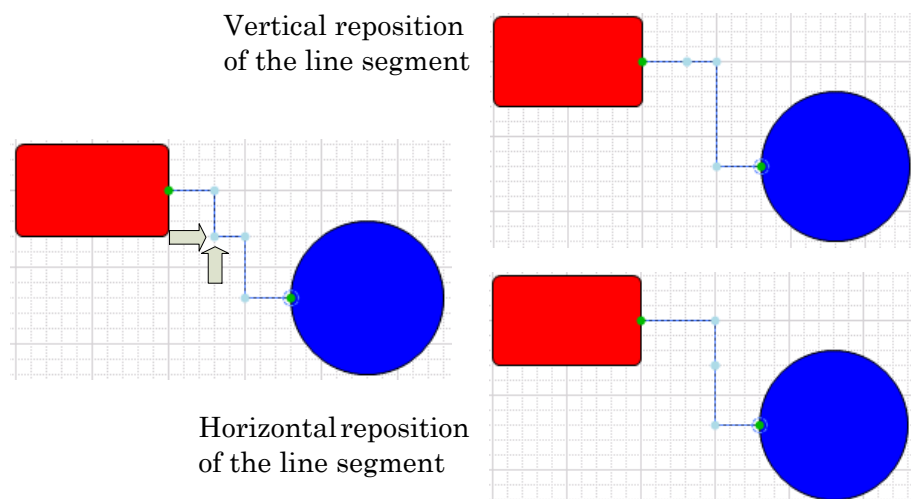


- 4 With your mouse key pressed, drag the line segment between the two adjacent control points to a new position to change the path of the connector.

Deleting a Control Point

An Angled connector is composed of two or more line segments with right angles between adjoining line segments. 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 134.

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

If you want to reverse a change you made in the Symbol Editor, use the undo function. After you undo a change, you can also redo that change by using the redo function.

You can undo one single change, or any number of changes that you have previously done. 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 a description of what the changes were.
- 2 Select a change from the list. 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. 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. For more information, see "Properties of Groups" on page 32.

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 press 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

- 1 On the canvas, select the group and also elements and groups that you want to add.
- 2 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.

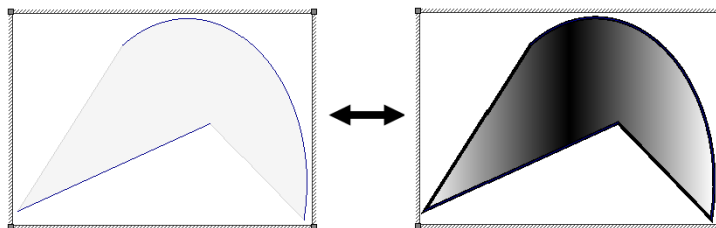
You can:

- 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.

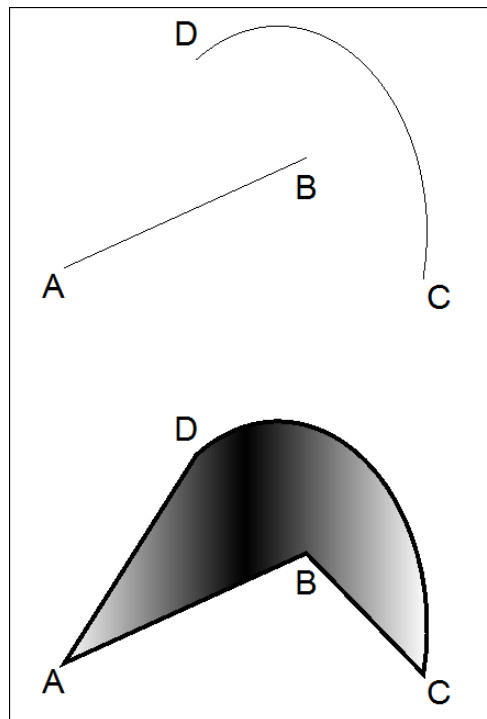


Creating a Path Graphic

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 158 and "Changing the Z-order of an Element in a Path Graphic" on page 159.

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 of 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.

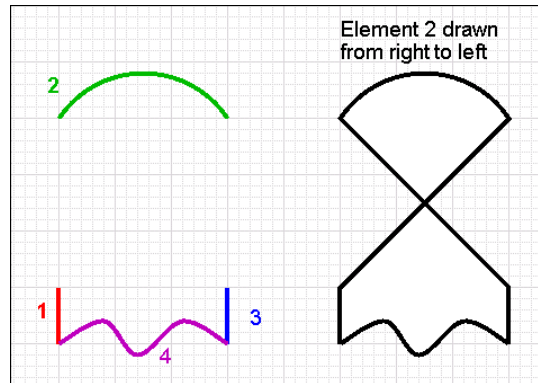
To edit control points of an element within a path graphic

- 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.



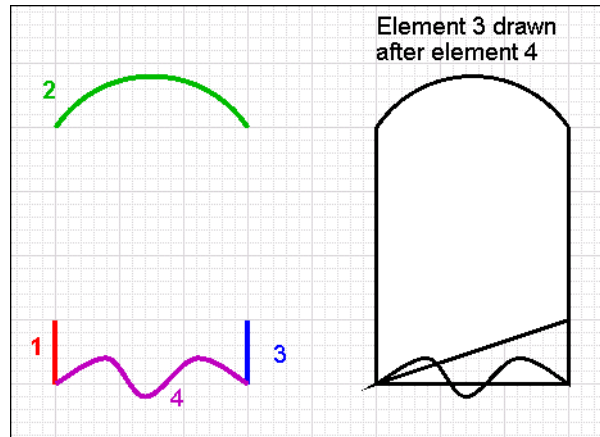
You can fix this by swapping the end points of the element where this appears.

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.
 - **Send 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 6

Editing Common Properties of Elements and Symbols

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.
- Tweaking 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 symbol.

For more information about properties that are specific to certain types of elements, see "Setting Symbol and Element-Specific Properties" on page 209.

Editing the Name of an Element

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 press 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 press 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 176.

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 176.

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 176.

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 176.
- 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 211.

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: Extreme 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 176.

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 176.

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 InTouch 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 176.

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:

| Click this command | To |
|---------------------------|---|
| Top Left | Align the text at the top left frame handle. |
| Top Center | Align the text at the top middle frame handle. |
| Top Right | Align the text at the top right frame handle. |
| Middle Left | Align the text at the middle left frame handle. |
| Middle Center | Align the text in the middle of the element. |
| Middle Right | Align the text at the middle right frame handle. |
| Bottom Left | Align the text at the bottom left frame handle. |
| Bottom Center | Align the text at the bottom center frame handle. |
| Bottom Right | 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 symbol, that text element is not updated if you change the source symbol text. For example, an embedded symbol contains a text graphic with the string "SomeTextHere". You substitute "SomeTextHere" with "MyText". You then changes the source symbol text from "SomeTextHere" to "Wonderware®". The text in the embedded symbol will still show "MyText".

To substitute strings in a symbol 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 symbol 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 ArchestrA Symbol 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 ArchestrA Symbol 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 188.

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

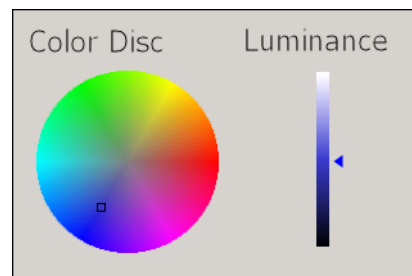
You can set a solid color from the standard palette using the **Solid Color** tab in the style selection dialog box. The standard palette is a set of 48 predefined colors you can use to quickly select a solid color.

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 179.

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 symbol 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 176.
- 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 176.

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 176.

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.

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 |

| Pattern | Options |
|----------|--|
| 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 176.
- 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.
 - 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. For more information, see "Properties of Groups" on page 32.

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. Click **OK**.

Note: You can also set the transparency level by changing the **Transparency** property in the Properties Editor.

Tweaking 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 press 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 ArcestrA, 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 symbols.

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.

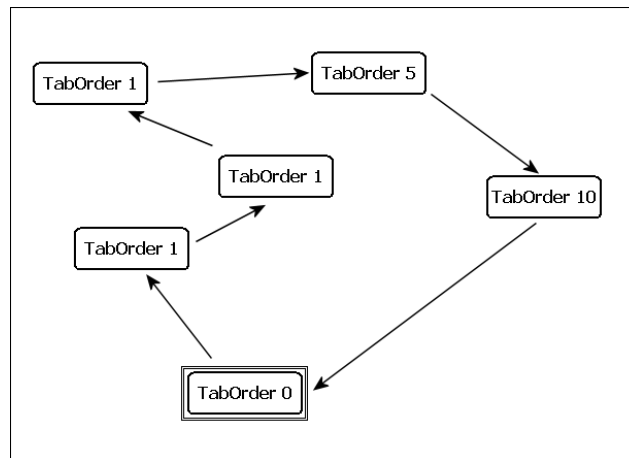
Editing the Tab Order of an Element

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 Symbol

You can configure the general properties of a symbol. The general properties determine the overall appearance and behavior of the symbol. You can:

- Add a meaningful description to your symbol.
- Enable anti-aliasing, or smoothing, for your symbol 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 symbol or display from a symbol. One example is a symbol 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 symbol

- 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 symbol

- 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 symbol

- 1 Click on the canvas so that no elements are selected.
- 2 In the Properties Editor, select True for the MultiplePopupsAllowed property.

Chapter 7

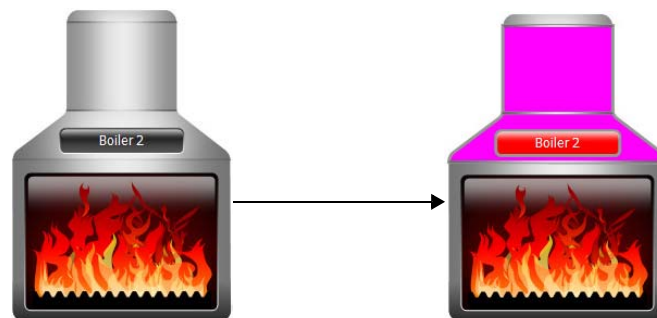
Working with Element Styles

An Element Style defines a set of visual properties that determine the appearance of text, lines, graphic outlines, and interior fill shown in ArcestrA Symbols or graphics. An Element Style applied to a symbol sets pre-configured visual property values that take precedence over a symbol's native visual properties.

Understanding Element Styles

Element Styles provide the means for developers to establish consistent visual standards in their ArcestrA applications. An Element Style can define the same visual properties of text, lines, fill, and outlines for all symbols or graphics that belong to an application.

Likewise, Element Styles can show the current status of an object represented by a symbol. For example, an Element Style animation can be applied to a symbol when an object transitions to an alarm state.



Galaxy Style Library

A set of Element Styles is provided in the predefined Galaxy Style Library. The predefined values of the Element Styles in this library can be changed. However, existing Element Styles cannot be renamed or deleted. Also, new Element Styles cannot be added to the library.

Visual Properties Defined by Element Styles

The following table lists the visual properties of graphic elements defined in an Element Style.

| Graphic Element | Element Properties |
|-----------------|---|
| Text | <ul style="list-style-type: none"> • Font family • Font size • Font style • Font color • Blink On/Off |
| Fill | <ul style="list-style-type: none"> • Fill color • Fill gradient • Fill pattern • Fill texture • Blink On/Off |
| Line | <ul style="list-style-type: none"> • Line pattern • Line weight • Line color • Blink On/Off |
| Outline | <ul style="list-style-type: none"> • Outline Show/Hide • Outline Pattern • Outline Weight • Outline Color • Blink On/Off |

An Element Style may not define every visual property. If a property value is not defined in an applied Element Style, the element's native style is used and can be changed. However, if an element's property value is defined in an applied Element Style, the element's native properties are disabled and cannot be changed.

Element Styles in Animations

You can configure an element or a group of elements with Boolean or truth table animations that determine whether Element Styles are applied based on evaluated conditions or expressions. See "Configuring an Animation Using Element Styles" on page 204.

Property Style Order of Precedence

To understand the behavior of an element's properties when an Element Style is applied, you should understand the order of precedence for the levels at which property styles are applied. For information about the property style order of precedence, see "Order of Precedence for Property Styles" on page 554.

Updating Element Styles at Application Run Time

You can update the Elements Styles applied to symbols or graphics included in a running application.

- Updating Element Styles from the IDE

When an application is deployed and updates were made to the applied Element Styles from the ArcestrA IDE, those updates will be propagated to the graphic elements in a running application without requiring WindowViewer to be closed and re-opened.

- Importing an updated Graphic Style Library

Importing an updated Graphic Style Library that includes different applied Element Styles will propagate those changes to graphic elements in a running application without requiring WindowViewer to be closed and re-opened.

Managing Element Styles

At the Galaxy level, you can import and export Galaxy Style Libraries containing custom Element Styles to applications running on other Galaxies. This section describes the tasks to create a set of custom Element Styles that can be used in other Galaxies.

Importing and Exporting Galaxy Style Libraries

You can import a Galaxy Style Library to load its Element Styles in a Galaxy. You can also modify Element Styles, and then export the Galaxy Style Library as custom user-defined Element Styles.

Optional Galaxy Style Library XML files are located in the
... \Program Files (x86) \ArcestrA \FrameWork \Bin \

AdditionalElementStyles folder. The names of the XML files suggest the primary color schemes of the Element Styles within each optional Galaxy Style Library. For more information about importing and exporting Galaxy Style Libraries, see Chapter 3 in the *Application Server User's Guide*.

Changing Visual Properties of an Element Style

You can modify the visual properties of any Element Style in the currently loaded Galaxy Styles Library. You modify properties by setting overrides on the **Element Styles** tab in the **Configure Galaxy Style Library** dialog box.

In the **Configure Galaxy Style Library** dialog box., you can:

- Modify the appearance of text by setting overrides for the text font, text size, text style, text color, and blinking.
- Modify the appearance of graphic fill by setting overrides for fill color and blinking.
- Override the appearance of the line pattern, weight, color, and blinking.
- Override the appearance of the outline line pattern, weight, color, and blinking.
- Preview the appearance of an Element Style.
- Reset Element Style visual properties to their default values.

To show the current Element Styles of a Galaxy

- 1 On the **Galaxy** menu, click **Configure** and click **Galaxy Style Library**. The **Configure Galaxy Style Library** dialog box appears.
- 2 Click the **Element Style** tab.

The **Element Styles** tab includes the following fields:

- The **Element Style Overrides** grid lists the Element Styles included in the library. An X within grid cells indicates style properties that have been overridden.
- The **Preview** field shows the appearance of an element when the current Element Style is applied.
- The **Reset to Default** button returns all modified Element Styles to their default values.
- The property tabs include related fields to set values for each property defined in the selected Element Style.

Overriding the Element Style Text Properties

You can modify an Element Style's text visual properties by setting alternative values for text font, text color, text style, and blink rate.

To change the appearance of text in an Element Style

- 1 On the **Galaxy** menu, point to **Configure**, and then click **Galaxy Style Library**. The **Configure Galaxy Style Library** dialog box appears.
- 2 Select an Element Style from the **Element Style Overrides** list.
- 3 Click the **Text (Ts)** tab.
- 4 To change the font, select **Font Override**, click the browse button, and select a font from the **Font** dialog box.
- 5 To override the font color:
 - a Select **Font Color Override**.
 - b Click the color box.
 - c Select a color from the **Select Font Color** dialog box.
- 6 To override the text blink behavior:
 - a Select **Blink**.
 - b Select a blinking speed from the **Speed** list.
 - c Click the color box to show the **Select Blink Color** dialog box.
 - d Select the color, gradient, pattern, and texture for the blink style.
- 7 Click **OK**.

Overriding the Element Style Fill Properties

You can modify an Element Style's fill visual properties by setting alternative values for fill color and blink rate.

To override the fill appearance of an Element Style

- 1 On the **Galaxy** menu, point to **Configure**, and then click **Galaxy Style Library**. The **Configure Galaxy Style Library** dialog box appears.
- 2 Select an Element Style from the **Element Style Overrides** list.
- 3 Click the **Fill** tab.
- 4 To override the fill style:
 - a Select **Fill Color Override**.
 - b Click the color box.
 - c Select a style from the **Select Fill Color** dialog box.

- 5 To override the fill blink behavior:
 - a Select **Blink**.
 - b Select a blink speed from the **Speed** list.
 - c Click the color box.
 - d Select a style from the **Select Fill Color** dialog box.
 - e Click **OK**.

Overriding the Element Style Line Properties

You can modify an Element Style's line visual properties by setting alternative values for line color, line pattern, and line weight

To override the line appearance of an Element Style

- 1 On the **Galaxy** menu, point to **Configure**, and then click **Galaxy Style Library**. The **Configure Galaxy Style Library** dialog box appears.
- 2 Select an Element Style from the **Element Style Overrides** list.
- 3 Click the **Line** tab.
- 4 To override the line pattern, select **Line Pattern Override** and select a line pattern from the adjacent list.
- 5 To override the line weight, select **Line Weight Override** and enter a new line weight in the adjacent box.
- 6 To override line color properties:
 - a Select **Line Color Override**.
 - b Click the color box.
 - c Select a color style from the **Select Line Color** dialog box.
- 7 To override the line blink behavior:
 - a Select **Blink**.
 - b Select a blinking speed from the **Speed** list.
 - c Click the color box.
 - d Select a style from the **Select Blink Color** dialog box.
- 8 Click **OK**.

Overriding the Element Style Outline Properties

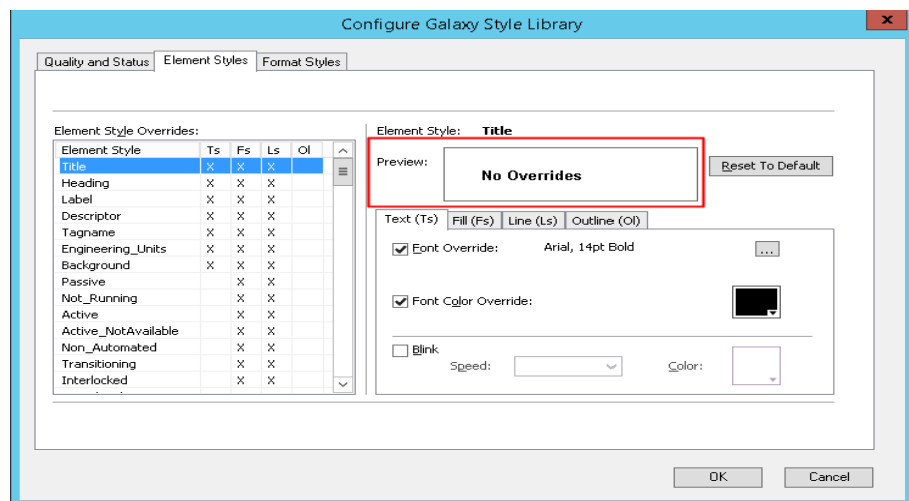
You can modify an Element Style's outline visual properties by setting alternative values for text font, text color, text style, and blink rate.

To override the outline appearance of an Element Style

- 1 On the **Galaxy** menu, point to **Configure**, and then click **Galaxy Style Library**. The **Configure Galaxy Style Library** dialog box appears.
- 2 Select an Element Style from the **Element Style Overrides** list.
- 3 Click the **Outline** tab.
- 4 Select **Show Outline**.
- 5 To set the line pattern, select **Line Pattern** and select a line pattern from the adjacent list.
- 6 To set the line weight, select **Line Weight** and type a line weight in the adjacent box.
- 7 To set the line style:
 - a Click the color box next to **Line Color**.
 - b Select a style from the **Select Line Color** dialog box.
- 8 To set the line blink behavior:
 - a Select **Blink**.
 - b Select a blinking speed from the **Speed** list.
 - c Click the color box.
 - d Select a blink style from the **Select Blink Color** dialog box.

Previewing an Element Style

The **Preview** area shows the appearance of an Element Style's current assigned property values.



To preview an Element Style

- 1 On the **Galaxy** menu, point to **Configure**, and then click **Galaxy Style Library**. The **Configure Galaxy Style Library** dialog box appears.
- 2 Select the **Element Styles** tab.
- 3 Select an Element Style from the **Element Style Overrides** list.
The **Preview** field updates to show the appearance of the selected Element Style.

Resetting an Element Style to Default Values

You can reset an Element Style to its original default property values.

Note: Resetting an Element Style resets visual properties to their original default values, not to any previous override settings.

To reset an Element Style to default values

- 1 On the **Galaxy** menu, point to **Configure**, and then click **Galaxy Style Library**. The **Configure Galaxy Style Library** dialog box appears.
- 2 Select the **Element Styles** tab.
- 3 Select one or more Element Styles from the **Element Style Overrides** list.
- 4 Click **Reset to Default**. All Element Style properties are reset to their default values.

Working with User-Defined Element Styles

The Galaxy Style Library includes a set of 25 user-defined Element Styles. User-defined Element Styles appear towards the bottom of the list of the **Element Style Overrides** field and are named User_Defined_01 to User_Defined_25.

Changing the Visual Properties of User-Defined Element Styles

All visual properties of user-defined Element Styles are initially set to default values. Visual properties can be individually configured for each user-defined Element Style by setting overrides for text, fill, line, and outline as other predefined Element Styles.

Renaming User-Defined Element Styles

Rename a style to provide a descriptive name, to provide a group of related styles names, or to suit any other specific application needs. You can rename user-defined Element Styles in Managed InTouch as well as in a InTouch Modern Application.

The renamed style will appear during configuration. For example, the renamed style will appear in the Graphic Editor when configuring an animation using Element Styles.

The renamed style will function the same during run time as it did prior to the name change.

To rename a user-defined Element Style

- 1 In the ArcestrA IDE, click **Galaxy** on the menu, then click **Configure**, then click **Galaxy Style Library**. The **Configure Galaxy Style Library** window appears.
- 2 On the Element Styles tab, navigate to the user-defined style you want to rename.
- 3 Use a mouse click, such as a "soft click" or a right-click, to select the style name and enable the text box for editing.
- 4 Enter your new style name.

Importing and Exporting User-Defined Element Styles

You can import and export your renamed user-defined Element Style as a normal part of the Galaxy Style Library. For more information, see "Importing and Exporting Galaxy Style Libraries" on page 195.

Applying Element Styles to Elements

You can apply Element Styles to one or more graphic elements. Unlike setting Element Style overrides that change the appearance of an Element Style's properties, applying an Element Style to a graphic element overrides the element's native properties.

Applying Element Styles to elements can help standardize the appearance of those elements in the Galaxy and show the current state of an object represented by a symbol or graphic. For more information, see "Changing Visual Properties of an Element Style" on page 196.

Using the Element Style List

The Symbol Editor menu bar contains an **Element Style** list to select an Element Style and apply it to a selected element of a symbol or graphic.

To apply an Element Style to a graphic element

- 1 Open the symbol or graphic in the Symbol Editor.
- 2 Select one or more elements from the graphic or symbol.
- 3 Select an Element Style from the **Element Styles** list to apply to the selected elements.

Using the Properties Grid

The Symbol Editor **Properties** view contains an **Element Style** Appearance item to select an Element Style and apply it to a selected element of a symbol or graphic.

To apply an Element Style from the Properties Editor

- 1 Open the symbol or graphic in the Symbol Editor.
- 2 Select one or more elements from the graphic or symbol.
- 3 In the **Appearance** category of the **Properties** Editor, select an Element Style from the **Element Style** list.

Using Format Painter

You can use the Symbol Editor's Format Painter to copy an Element Style from one graphic element to another.

To apply an Element Style using Format Painter

- 1 Open a symbol or graphic in the Symbol Editor.
- 2 Select the element with the Element Style you want to copy.
- 3 On the **Edit** menu, click **Format Painter**. The pointer appears as the Format Painter cursor.
- 4 Select the elements you want to apply the Element Style to. The Element Style is applied to the clicked element.

Clearing an Element Style

When an Element Style is applied to an element, you cannot edit the element's styles that are controlled by the applied Element Style. However, you can clear the application of the Element Style so that all of the styles can be edited.

To clear an Element Style

- 1 Select the element.
- 2 Select **None** in the Element Style list.

Selecting an Element Style as a Default for a Canvas

You can select an Element Style at the canvas level. The selected Element Style is applied to any graphic element or groups that you create on the canvas.

Applying Element Styles to Groups of Elements

You can apply an Element Style on a group of elements in the same way that you apply an Element Style to an element. However, the group's run-time behavior must be set to `TreatAsIcon`.

Setting a Group's Run-time Behavior to `TreatAsIcon`

To apply an Element Style to a graphic element group, the group's **`TreatAsIcon`** property must be set to `True`. Otherwise, the Element Style lists are disabled when an element group is selected.

To set a group's `TreatAsIcon` property to true

- 1 Select the element group to which the Element Style will be applied.
- 2 On the **Properties** menu, click **Run-time Behavior** and click **`TreatAsIcon`**.
- 3 Select **`True`** from the drop-down list.

Understanding Element Style Behavior with a Group of Elements

- The Element Style applied to a group has higher precedence than the property styles applied to individual graphic elements in the group.
- If the Element Style applied to a group of elements has undefined property styles, then the element continues to use its Element Style or element-level settings for undefined property styles.
- If the Element Style that is applied to a group of elements has defined property styles, then those property styles override the property styles defined at the element level for elements in the group.
- An Element Style cannot be applied to a nested element group.
- If you add an element to a group that has a group-level Element Style applied, the group Element Style is applied to it.

Configuring an Animation Using Element Styles

You can configure an element or a group of elements with a:

- Boolean animation that applies Element Styles based on a binary True/False condition.
- Truth table animation that applies Element Styles based on a range of possible values.

The truth table animation that applies Element Styles:

- Associates expressions of any data type supported by Application Server or InTouch to an Element Style.
- Defines as many conditions as required and applies a separate Element Style for each condition
- Defines the conditions to apply an Element Style by specifying a comparison operator (=, >, >=, <, <=) and a breakpoint, which itself can be a value, an attribute reference, or an expression.
- Arranges conditions in the order that Element Styles are processed.

Configuring a Boolean Animation Using Element Styles

You can configure an element or a group of elements with a Boolean animation that uses only two Element Styles.

To configure an element or a group of elements with an Element Style that uses Boolean animation

- 1 Open the symbol or graphic in the IDE Symbol Editor.
- 2 Select the element or element group.
- 3 On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
- 4 Click the **Add** icon and select **Element Style**. The Element Style animation is added to the Animation list and the **Element Style** state selection panel appears.
- 5 Click the **Boolean** button. The **Boolean Element Style** configuration panel appears.
- 6 In the **Boolean** text box, enter a Boolean numeric value, attribute reference, or an expression.
- 7 Clear **ElementStyle** in the **True, 1, On** area or **False, 0, Off** area if you do not want a different Element Style for the true or false condition than the default Element Style that is shown in the **Element Style** list.
- 8 In the **True, 1, On** area, select the Element Style in the list to use when the expression is true.
- 9 In the **False, 0, Off** area, select the Element Style in the list to use when the expression is false.
- 10 Click **OK**.

Configuring a Truth Table Animation with Element Styles

You can configure an element or a group of elements with a Truth Table animation that selects multiple Element Styles based on a set of evaluated values or expressions.

To configure an element or a group of elements with an Element Style that uses Truth Table animation

- 1 Open the symbol or graphic in the IDE Symbol Editor.
- 2 Select the element or group.
- 3 On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
- 4 Click the **Add** icon and select **Element Style**. The Element Style animation is added to the Animation list and the **Element Style** state selection panel appears.

- 5 Click the **Truth Table** button. The **Truth Table Element Style** configuration panel appears. The Element Style that is applied to the element is shown in the **Element Style** list at the bottom of the panel.
- 6 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.
- 7 If the data type of the expression is string or internationalized string, you can specify to ignore the case by selecting **Ignore Case**.
- 8 In the **Truth Table**, select the **Element Style** check box and select the Element Style for one of the conditions to be defined in the truth table.
- 9 In the **Operator** column, select a comparison operator.
- 10 In the **Value or Expression** column, type a value, attribute reference, or expression.
- 11 To add other conditions:
 - a Click the **Add** icon. An additional condition is added to the truth table.
 - b Select the **Element Style** check box, select the Element Style for the condition, select an operator, and enter the condition value or expression.
- 12 After adding all truth table conditions, click **OK**.

Truth Table animation is typically used to set Element Styles to the different states of an object. For example, you can set Truth Table conditions to show different Element Styles that represent the following alarm conditions:

- When the attribute TankLevel_001.PV is 0 then no Element Style is applied.
- When the attribute TankLevel_001.PV is less than 20, then the Element Style is Alarm_Minor_Dev.
- When the attribute TankLevel_001.PV is greater than the attribute Standards.TankMax then the Element Style is Alarm_Major_Dev.

Deleting a Condition from an Animation Truth Table

You can delete a condition from an animation Truth Table to remove the associated Element Style from the animation.

To delete a condition from a Truth Table animation that uses Element Styles

- 1 Open the **Edit Animations** dialog box, **Truth Table Element Style** panel.
- 2 Select the condition you want to delete.
- 3 Click the **Remove** icon. The condition is removed.

Changing the Processing Order of Element Styles in a Truth Table Animation

You can change the processing order of Element Styles by moving the conditions up or down in the Truth Table list. The Element Style at the top of the Truth Table list is processed first. The remaining Element Styles are processed in order based on their position from the top of the list.

To change the processing order of Element Style conditions

- 1 Open the **Edit Animations** dialog box, **Truth Table Element 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.

Chapter 8

Setting Symbol and Element-Specific Properties

You can configure symbol-specific and element-specific properties. For properties that are common to all or most elements, see "Editing Common Properties of Elements and Symbols" on page 163.

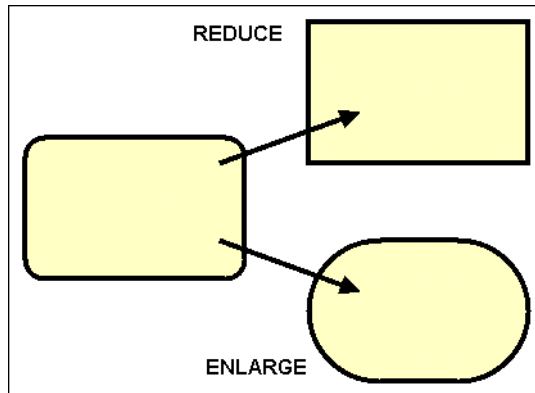
You can configure:

- General properties of a symbol.
- 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 symbol containing rounded rectangles is embedded into an InTouch window and resized, the radius is not affected. This can have adverse affects on the graphic representation of your symbol.

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 press 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 176.

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 215.

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

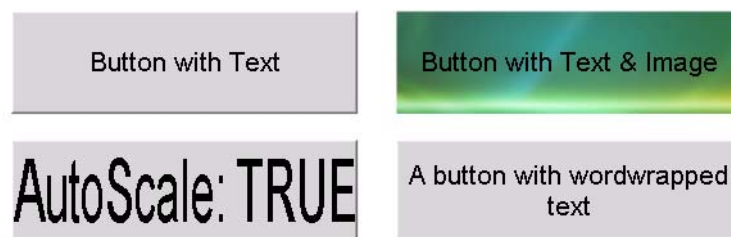
- 1 Select the image element with the image you want to change.
- 2 On the **Edit** menu, click **Select Image**. The **Open** dialog box appears.
- 3 Browse to and select an image file, and then click **Open**. The image is loaded into the image frame.

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 ArcestrA symbols. 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 ArcestrA Symbols.

- 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 symbol.
- 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 Ctrl key.
- 4 Move the mouse over the control point you want to remove. The pointer appears as a pen with a minus symbol.
- 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 220.

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 the angle to be multiples of 15 degrees, press and hold the Shift key.
- 4 If you want 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 Symbol 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 symbol to show non-good status and quality of attributes in different ways:

- A status element shows a specific icon depending on the quality and status of configured attributes or elements. See "Using Status Elements" on page 221.
- The text, fill, or line appearance of elements is overridden depending on the quality and status of the attributes they reference. See "Overriding Element Appearance Depending on Quality and Status of its Attributes" on page 224.
- Elements are drawn with an outline depending on the quality and status of the attributes they reference. See "Overriding Element Appearance Depending on Quality and Status of its Attributes" on page 224.

Using Status Elements

Status elements show a specified symbol depending on the quality and status of:

- Attributes configured for specific animated elements.
- One or more specified attributes.

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.

Drawing the Status Element on the Canvas

You can easily place a status element on the canvas to show an icon that indicates quality and status of attributes contained in selected animated elements and/or specified attributes.

You do this as you would with any other element. For more information, see "Drawing and Dragging Elements" on page 110.

Configuring the Status Element

You can associate the status element with:

- Animated elements that use attributes that provide the quality and status that is to be monitored.
- Attributes that provide the quality and status that is to be monitored.

In both cases, the appearance is set by the settings in the **Quality and Status** tab of the **Configure Galaxy Style Library** dialog box.

For more information on how to configure this animation, see "Configuring Animation for a Status Element" on page 341.

Setting the Appearance of a Status Element

You can set the appearance of a status element depending on the quality and status of its referenced attributes and/or attributes used in its referenced elements.

You can also preview the appearance of a status element. For more information, see "Previewing all Status Appearances" on page 227.

You can reset the appearance of a status element to its default. For more information, see "Resetting an Override Appearance to its Default" on page 228.

To set the default appearance of a status element

- 1 On the **Galaxy** menu, point to **Configure**, and then click **Galaxy Style Library**. The **Configure Galaxy Style Library** dialog box appears with the **Quality and Status** tab.
- 2 Click the **Quality and Status** tab.
- 3 Select **Enable Quality and Status Display**.
- 4 Select a status or quality from the **Status Style Overrides** list.
- 5 Click the **Status (St)** tab to show the default visual properties of status or qualities.
- 6 To set the line style of the frame around a status or quality element:
 - a Select **Line**.

- Click **TopRight** to align at the top right corner.
- Click **BottomRight** to align at the bottom right corner.

14 Click **OK**.

Overriding Element Appearance Depending on Quality and Status of its Attributes

You can configure any animated element to appear differently depending on the quality and status of its associated attributes.

For animated elements, you can:

- Override the appearance of the text font, style, and blinking.
- Override the appearance of the fill style and blinking.
- Override the appearance of the line style, weight, pattern, and blinking.
- Preview all status appearances in one dialog box.
- Reset the status appearances to their defaults.
- Use an outline to indicate a specified status or quality.

Note: Instead of overriding the appearance of elements on the canvas, you can use a status element. The status element shows an icon representing quality and status of monitored attributes.

You can configure the appearance overrides and Status element overrides in the **Configure Galaxy Style Library** dialog box, which you access by selecting the **Configure** option of the **Galaxy** menu.

Overriding the Text Appearance of Elements to Indicate Non-Good Status or Quality

You can configure the Galaxy so that the text appearance of animated elements with attributes that have non-good status or quality are overridden with a specific text appearance.

To override the text appearance of an element specified by a status element

- 1** On the **Galaxy** menu, point to **Configure**, and then click **Galaxy Style Library**. The **Configure Galaxy Style Library** dialog box appears.
- 2** Click the **Quality and Status** tab.
- 3** Select **Enable Quality and Status Display**.
- 4** Select a status or quality from the **Status Style Overrides** list.
- 5** Click the **Text (Ts)** tab.

- 6 To override the font, select **Font Override**, click the browse button and select a font from the **Font** dialog box.
- 7 To override the font style:
 - a Select **Font Color Override**.
 - b Click the color box.
 - c Select a text color from the **Select Font Color** dialog box. For more information, see "Setting Style" on page 176.
- 8 To override the text blink behavior:
 - a Select **Blink**.
 - b Select a blinking speed from the **Speed** list.
 - c Click the color box.
 - d Select a text blink color from the **Select Blink Color** dialog box. For more information, see "Setting Style" on page 176.
- 9 Click **OK**.

Overriding the Fill Appearance of Elements to Indicate Non-Good Status or Quality

You can configure the Galaxy so that the fill appearance of animated elements with attributes that have non-good status or quality are overridden with a specific fill appearance.

To override the fill appearance of an element specified by a Status element

- 1 On the **Galaxy** menu, point to **Configure**, and then click **Galaxy Style Library**. The **Configure Galaxy Style Library** dialog box appears.
- 2 Click the **Quality and Status** tab.
- 3 Select **Enable Quality and Status Display**.
- 4 Select a status or quality from the **Status Style Overrides** list.
- 5 Click the **Fill (Fs)** tab.
- 6 To override the fill style:
 - a Select **Fill Color Override**.
 - b Click the color box.
 - c Select a fill color from the **Select Fill Color** dialog box. For more information, see "Setting Style" on page 176.

- 7 To override the fill blink behavior:
 - a Select **Blink**.
 - b Select a blinking speed from the **Speed** list.
 - c Click the color box.
 - d Select a fill blink color from the **Select Blink Color** dialog box. For more information, see "Setting Style" on page 176.

Overriding the Line Appearance of Elements to Indicate Non-Good Status or Quality

You can configure a Galaxy so that the line appearance of animated elements with attributes that have non-good status or quality are overridden with a specific line appearance.

To override the line appearance of elements specified by a Status element

- 1 On the **Galaxy** menu, point to **Configure**, and then click **Galaxy Style Library**. The **Configure Galaxy Style Library** dialog box appears.
- 2 Click the **Quality and Status** tab.
- 3 Select **Enable Quality and Status Display**.
- 4 Select a status or quality from the **Status Style Overrides** list.
- 5 Click the **Line (Ls)** tab.
- 6 To override the line pattern, select **Line Pattern Override** and select a line pattern from the adjacent list.
- 7 To override the line weight, select **Line Weight Override** and type a new line weight in the adjacent box.
- 8 To override the line color:
 - a Select **Line Color Override**.
 - b Click the color box.
 - c Select a line color from the **Select Line Color** dialog box. For more information, see "Setting Style" on page 176.
- 9 To override the line blink behavior:
 - a Select **Blink**.
 - b Select a blinking speed from the **Speed** list.
 - c Click the color box.
 - d Select a line blink color from the **Select Blink Color** dialog box. For more information, see "Setting Style" on page 176.
- 10 Click **OK**.

Adding Outlines to Elements to Indicate Non-Good Status or Quality

You can configure the Galaxy so that animated elements with attributes that have non-good status or quality are shown with an outline.

To add outlines to elements to indicate non-good status or quality

- 1 On the **Galaxy** menu, point to **Configure**, and then click **Galaxy Style Library**. The **Configure Galaxy Style Library** dialog box appears.
- 2 Click the **Quality and Status** tab.
- 3 Select **Enable Quality and Status Display**.
- 4 Select a status or quality from the **Status Style Overrides** list.
- 5 Click the **Outline (OI)** tab.
- 6 Select **Show Outline**.
- 7 To set the line pattern, select **Line Pattern** and select a line pattern from the adjacent list.
- 8 To set the line weight, select **Line Weight** and type a line weight in the adjacent box.
- 9 To set the line style:
 - a Click the color box next to **Line Color**.
 - b Select a line color from the **Select Line Color** dialog box. For more information, see "Setting Style" on page 176.
- 10 To set the line blink behavior:
 - a Select **Blink**.
 - b Select a line blink speed from the **Speed** list.
 - c Click the color box.
 - d Select a line blink color from the **Select Blink Color** dialog box. For more information, see "Setting Style" on page 176.

Previewing all Status Appearances

You can preview the appearance of all status overrides by showing the **Status Legend** dialog box.

To preview all override appearances

- 1 On the **Galaxy** menu, point to **Configure**, and then click **Galaxy Style Library**. The **Configure Galaxy Style Library** dialog box appears.
- 2 Click the **Quality and Status** tab.

- 3 Click **Preview Legend**. The **Status Legend** dialog box appears.
- 4 Click **Close**.

Resetting an Override Appearance to its Default

For any status, you can reset the default appearance:

- Text, fill, and line overrides.
- Outline settings.
- Status element settings.

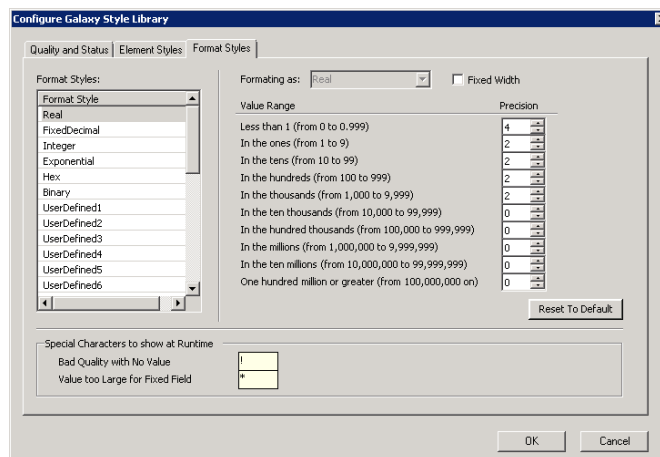
To reset a status or quality to its default appearance

- 1 On the **Galaxy** menu, point to **Configure**, and then click **Galaxy Style Library**. The **Configure Galaxy Style Library** dialog box appears.
- 2 Click the **Quality and Status** tab.
- 3 Select a status or quality from the **Status Style Overrides** list.
- 4 Click **Reset to Default**. All text, fill, and line overrides, status element icons, and outline settings are reset to their defaults.

Setting Global Number Styles

The **Configure Galaxy Style Library** dialog box includes the **Format Styles** tab. **Format Styles** provides options to individually configure Galaxy-wide styles for common types of numbers used in industrial applications.

Each global number style is assigned a unique name, which cannot be changed. A number style can be applied by name in design time and run time for an analog data type in User Input and Value Display animations. Also, grouped elements support global number styles.



The **Format Style** list includes a set of styles that can be individually configured to create customized number styles. The Real number format is the default for the user defined styles.

The **Format Styles** dialog box also includes fields to specify characters that appear during run time that indicate bad quality data or data that exceeds a fixed width field.

Based on the selected number style, the following fields appear on the **Format Styles** dialog box to change the default properties.

- **Fixed Width**

Appears for every number format style. When selected, the length of a number cannot exceed the text length of the text element (Text, TextBox, or Button) in design time. Numbers that exceed design time text length will show the special character specified in the **Value too large for Fixed Field**.

For the Real number style, the length of the fractional part of the number is truncated to fit the design time length. If the length of the number is still too large after removing the entire fractional part, then the number will show the special character specified in the **Value too large for Fixed Field**.

- **Precision**

Appears for the **Real**, **FixedDecimal**, and **Exponential** number styles. **Precision** sets the possible number of digits in the fractional part of a number to the right of the decimal point and can be set from 0 to 8.

- **Bits From and To**

Appear for the **Hex** and **Binary** number formats. **Bits From** sets the starting bit position (0-31) 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.

Configuring Global Number Styles

You can configure a global number style by changing the values assigned to the properties for each type of number.

To configure a global numeric format style

- 1 On the **Galaxy** menu, point to **Configure**, and then click **Galaxy Style Library**. The **Configure Galaxy Style Library** dialog box appears.
- 2 Click the **Format Styles** tab.

- 3 Select a numeric format style from the **Format Styles** list.

The **Format Styles** dialog box updates to show fields for **Fixed Width**, **Precision**, **Bits From**, or **To** based on the selected format style.

- 4 Update the fields shown for the selected format style.
- 5 Click **OK** to save the changes to the global number style.

Working with User-defined Global Number Styles

The Galaxy Style Library includes a set of 25 user-defined number styles. User-defined styles appear towards the bottom of the list of the **Format Styles** list and are named UserDefined1 to UserDefined25.

Renaming User-defined Global Number Styles

Rename a style to provide a descriptive name, to provide a group of related styles names, or to suit any other specific application needs. You can rename user-defined Format Styles in Managed InTouch as well as in a InTouch Modern Application.

The renamed style will appear during configuration. For example, the renamed style will appear in the Graphic Editor when configuring an animation using Number Styles.

The renamed style will function the same during run time as it did prior to the name change.

To rename a user-defined element style

- 1 In the ArcestrA IDE, click **Galaxy** on the menu, then click **Configure**, then click **Galaxy Style Library**. The **Configure Galaxy Style Library** window appears.
- 2 On the **Format Styles** tab, navigate to the user-defined style you want to rename.
- 3 Use a mouse click, such as a "soft click" or a right-click, to select the style name and enable the text box for editing.
- 4 Enter your new style name.

Importing and Exporting User-Defined Global Number Styles

You can import and export your renamed user-defined Number Style as a normal part of the Galaxy Style Library. For more information, see "Importing and Exporting Galaxy Style Libraries" on page 195.

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.

ArchestrA 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 InTouch WindowViewer.

Numeric formatting by regional locale applies only to ArchestrA graphic number displays and input numbers included as part of InTouch managed and Modern applications.

Important: Native InTouch user input or value display animations do not support numeric formatting by regional locale.

Numeric formatting by regional locale can be applied to InTouch managed or Modern applications:

- User Input animation
- Value Display animation
- Tooltip animation
- Windows Client Controls (RadioBox, ComboBox, and ListBox)
- SecuredWrite dialog
- VerifiedWrite dialog
- SignedWrite dialog

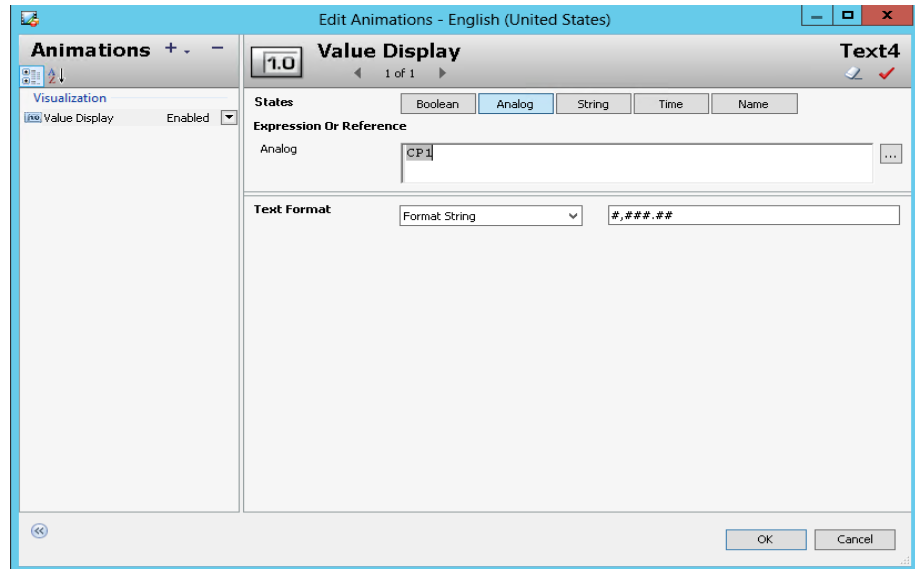
Design Time Considerations for Numeric Formatting

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 WindowViewer
- Select the **Regional Settings in ArchestrA Graphics** WindowViewer option

Enter Input Numbers in U.S. Format

The ArcestrA Graphics Editor enables you to specify a number format during design time using a format string for User Input and Value Display animations.



Both animation types include 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 from WindowViewer 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.

User Input and Value Display animation displayed in German number format during run time

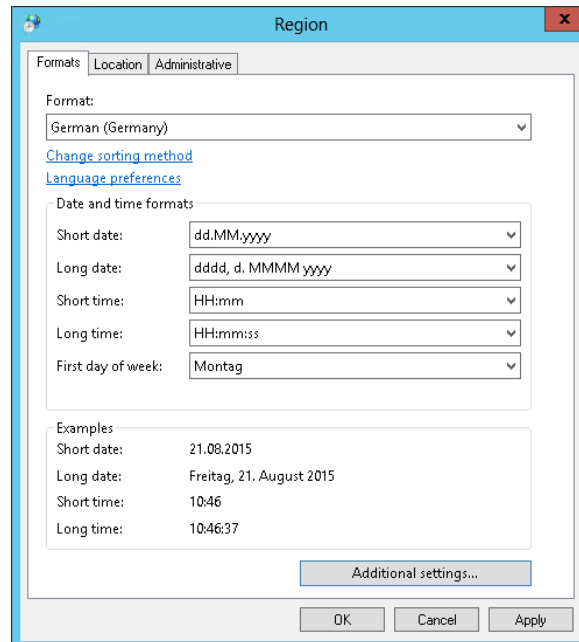
| Design-Time format string in US format | Format String | User Input | Value Display |
|--|---------------|-------------|---------------|
| | #,###.## | 8.456,47 | 8.456,47 |
| | #,###,###.## | 8.456,47 | 8.456,47 |
| | ####.## | 8456,47 | 8456,47 |
| | #.###,## | 8456.470000 | 8456.470000 |

Incorrect non-US number format Invalid numbers

In the example above, the design-time numeric format strings within the red 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 Running WindowViewer

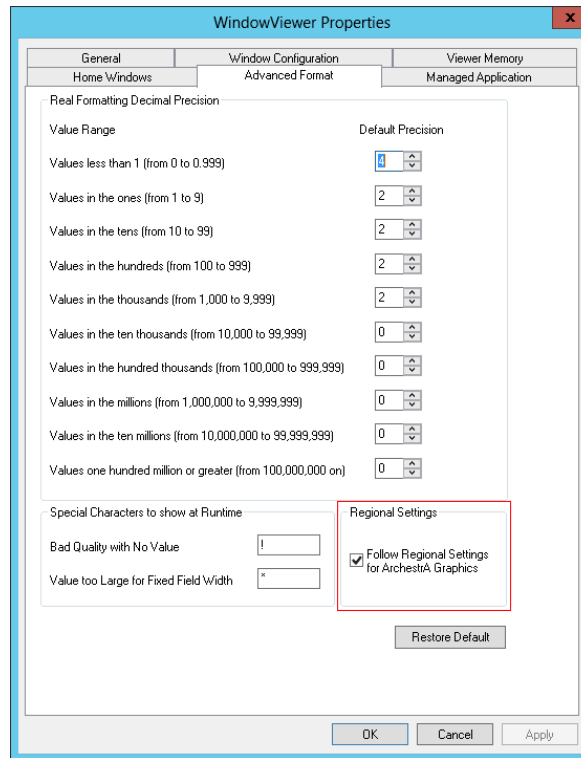
To enable numeric formatting by regional locale, the computer on which WindowViewer is installed must have its region set to the country in which you want ArcestrA graphic numbers to be formatted.



The Region setting is accessible from the Windows Control Panel. If you want to display ArcestrA graphic numbers in a non-U.S. format, select the **Formats** tab and select a country in the **Format** field.

Select the Regional Settings WindowViewer Option

InTouch WindowMaker includes a WindowViewer Advanced Format **Regional Settings** configuration option.



To enable numeric formatting by regional locale, the **Regional Settings** option must be selected during design time to format ArcestraA Graphic numbers to the country selected in the **Region** setting. By default, the **Regional Settings** option is disabled.

Note: WindowViewer checks the OS Regional Settings only on startup. This means that you may need to restart WindowViewer if you do either of the following:

- 1) Select the **Regional Settings** option while WindowViewer is running.
- 2) Change the OS Regional Settings while WindowViewer is running with the **Regional Settings** option selected.

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:

| Numeric Format Styles | Value Display |
|-----------------------|---------------|
| 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, the Arcestra IDE and WindowViewer provide software keypads that 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 image shows a numeric keypad interface. At the top, it displays 'Current Value: 8 456.47'. Below that, it shows 'Minimum Value: 500.5' and 'Maximum Value: 9 000.5'. A 'New Value' input field contains '8 456.47'. The keypad consists of a 4x4 grid of buttons: the first three columns contain digits 7-9, 4-6, 1-3, and a decimal point; the fourth column contains a left arrow, a period, a comma, and a space. At the bottom 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 your InTouch managed or Modern applications.

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 `WindowViewer` computer's 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 symbol:

- 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, clicking on the canvas to position it and, with exception of the calendar control, dragging a rectangle to set the size.

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 176. 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 176. 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> |
| DateTime 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> |

For more information about scripting, see "Adding and Maintaining Symbol Scripts" on page 369.

For more information about the value property, see "Alphabetical List of Properties" on page 505.

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

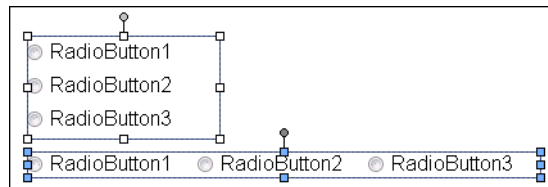
You can set the 3D appearance of a radio button group control. This affects how the option circles appear.

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 the Galaxy Browser. For more information about scripting, see "Adding and Maintaining Symbol Scripts" on page 369.

For more information about the properties, see "Alphabetical List of Properties" on page 505.

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.

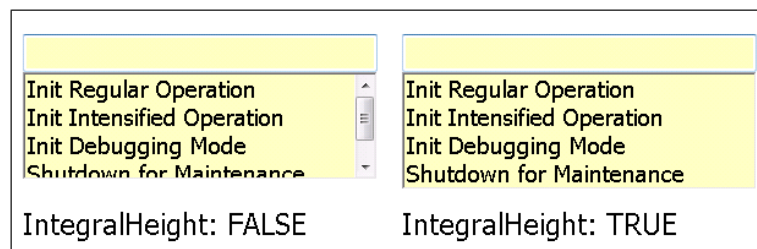
Typically you set the drop-down list width greater than the width of the combo box on the canvas. If you set the drop-down list width smaller than the combo box control width on the canvas, the drop-down list is the same width as the combo box control.

To set the width of the combo box drop-down list

- 1 Select the combo box control.
- 2 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

- 1 Select the combo box control.
- 2 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

- 1 Select the combo box control.
- 2 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 the Galaxy Browser. For more information about scripting, see "Adding and Maintaining Symbol Scripts" on page 369.

For more information about the properties, see "Alphabetical List of Properties" on page 505.

Configuring Calendar Controls

You can use the Calendar control to select a date from one or more monthly calendar sheets.

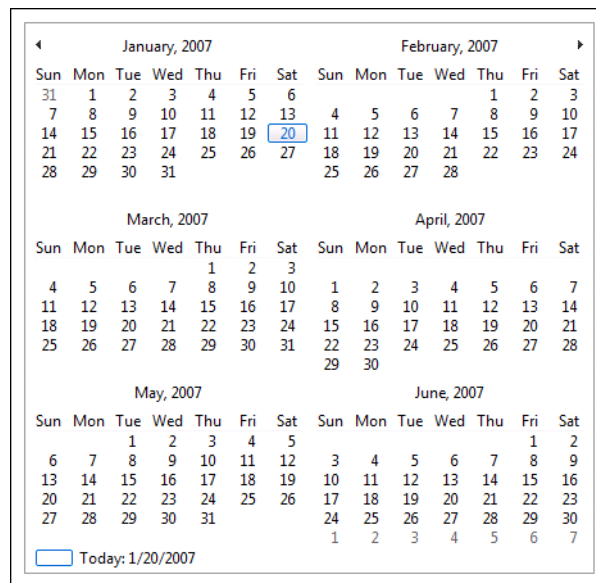
You can:

- 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.



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.

When you change the title fill color, this 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 background color of a Calendar control

- 1 Double-click the Calendar control to show the Properties Editor.
- 2 In the **Calendar Colors** field, click **Title Background**.

The **Select Title Background Color** dialog box appears. For more information, see "Setting Style" on page 176.

- 3 Select a color and click **OK**.

The background color of the Calendar control title bar changes to the color you selected.

To change the title text color of a Calendar control

- 1 Double-click the Calendar control to show the Properties Editor.
- 2 In the **Calendar Colors** field, click **Title Foreground**.
The **Select Title Foreground Color** dialog box appears. For more information, see "Setting Style" on page 176.
- 3 Select a color and click **OK**.
- 4 The text color of the Calendar control title bar changes to the color you selected.

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 176.
- 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 IntegralHeight 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.

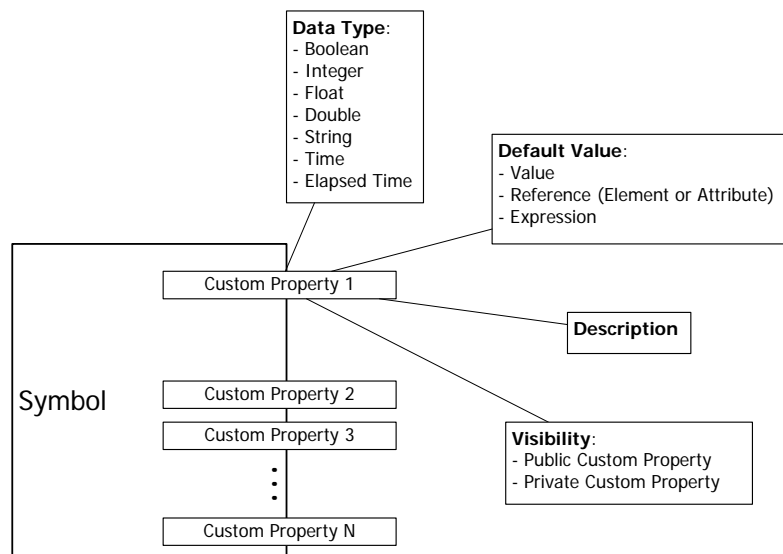
These properties are available when you browse for a List Box control in the Galaxy Browser. For more information about scripting, see "Adding and Maintaining Symbol Scripts" on page 369.

For more information about the properties, see "Alphabetical List of Properties" on page 505.

Chapter 9

Using Custom Properties

You can configure and use custom properties to extend the functionality of symbols and use them in combination with InTouch tags. You can use binding with custom properties to dynamically change the reference of a custom property.



About Custom Properties

You use custom properties to extend the standard properties of a symbol or an embedded symbol. 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 ArchestrA Symbol to InTouch tags.

Managing Custom Properties

You manage all custom properties of a symbol 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 symbol.

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 symbol and the custom property in the header of the dialog box.

Note: If the symbol includes an embedded symbol, the name of the custom property cannot be the same as the name of the embedded symbol or of an element of the embedded symbol.

Note: If the symbol 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 255.
- 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 symbol.
- 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 you the symbol name, for example Symbol_001, on the right and the custom property name on the left, for example MyCustomProperty. It can be accessed from scripting as 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 | Symbol |
|--------------|---|
| 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 symbol is embedded into another symbol, 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 symbol 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 symbol if the symbol is embedded.
 - Click **Private** if you want the custom property to be hidden and no reference be made to it outside of the defining symbol.
- 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.

Note: If the symbol includes an embedded symbol, the name of the custom property cannot be the same as the name of the embedded symbol or of an element of the embedded symbol.

Note: If the symbol 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 symbol directly to external sources by:

- Configuring AutomationObjects that point to external sources and then point the custom property at the corresponding attribute reference.
- Configuring a special InTouch reference syntax in the **Default Value** box. When you embed the symbol into an InTouch window, the referenced InTouch tags connect to the tags of the InTouch HMI.

Note: ArcestrA custom properties referencing InTouch tags which have hyphens in their names will not work in run time. For example, "InTouch: TAG-1" will not work in run time.

For more information, see "Connecting Animations with Custom Properties" on page 276, "Connecting Animations with InTouch Tags" on page 277 and the *InTouch HMI and ArcestrA Integration Guide*.

Overriding Custom Properties

You can override the custom property default values of embedded symbols within symbols in the ArcestrA Symbol Editor or InTouch WindowMaker.

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 symbols in other symbols and in InTouch WindowMaker.

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.

Examples of Using Custom Properties

Possible uses for using custom properties are:

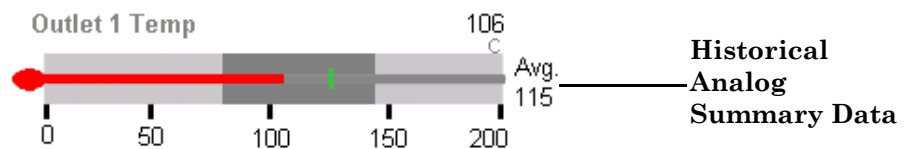
- A "TankLevel" custom property of type Writable Attribute can be given a value of "me.pv".
- A "MaxFillLevel" custom property of type Expression can be given a value of "Me.MaxCapacity - 200".

A more extensive example on how to use custom properties in embedded symbols in an InTouch window can be found in the *InTouch HMI and ArcestrA Integration Guide*.

Using Custom Properties to Show Historical Summary Data

You can add a custom property to reference historical data collected over a specified period during run time. The Historian can transform this data to create a set of analog or state statistics that can be shown by ArcestrA graphic animation during run time.

For example, consider an example of a temperature meter symbol that shows an optimal temperature range and you would like to know what the average temperature has been over the last 15 minutes. You can add a Value Display animation that shows the average temperature derived from analog temperature data saved in the Historian and referenced by a custom property.



Analog Statistical Summary Data

A custom property analog reference can subscribe to statistics from analog data collected over a defined summary period and saved to the Historian. The following table lists the analog historical statistical data that can be specified for a custom property.

| Analog Statistical Data | Description |
|--------------------------------|---|
| Average | A time-weighted average calculated from values within a summary period. The average is calculated at the end of the summary period. |
| Count | A value count calculated from values within a summary period. The count is calculated at the end of the summary period. |
| First | The first value that occurs within a summary period based on the actual timestamp within the summary period. |
| Integral | An integral value calculated from values within a summary period. The integral is calculated at the end of the summary period. |
| Maximum | The first maximum value that occurs within a summary period. |
| Minimum | The first minimum value that occurs within a summary period. |
| PercentGood | The ratio of labeled "good" quality data to all data within the summary period. The ratio is expressed as a percentage in the range 0 to 100. PercentGood is calculated at the end of the summary period. |
| StdDev | Time weighted standard deviation calculated from values within a summary period. The value is calculated using time weighted sums (Integrals) and time weighted sums of squares (IntegralOfSquares) values. |
| Last | The last value that occurred in the summary period based on the actual timestamp within the summary period. |

State Statistical Summary Data

The Wonderware Historian stores and retrieves values, where every value gets stored with some timestamp and associated quality. This triplet of value, timestamp, and quality is called VTQ and constitutes the smallest addressable piece of data in the Wonderware Historian data model.

State summary statistics summarize the states of a value. Three different state value data types are possible: analog (integer), string, and Null.

A custom property state reference can subscribe to state statistics from the Historian as static text, an expression or reference, an aggregate function name, minutes, and state value.

The Historian returns the VTQ for one cycle of a specified state. The quality returned is always OpcQuality. The time returned is always the summary period start time. Value and Time differ based on the aggregate function.

The following table lists state historical statistical data that can be specified for a custom property.

| State Statistical Data | Description |
|-------------------------------|---|
| Average | Average time a state occurred and completed within a summary period. A partial state within a summary period is ignored for an average calculation. (StateTimeAvgContained) |
| Minimum | Minimum time a state occurred and completed within a summary period. A partial state is ignored. (StateTimeMinContained) |
| Maximum | Maximum time a state occurred within a summary period. (StateTimeMax) |
| Count | Number of times a state occurred and completed within a summary period. A partial state is not counted. (StateCountContained). |
| Percent | Percentage of the summary period that a state occurred. (StateTimePercent) |
| Total | Total time a state occurred within a summary period. (StateTimeTotal) |

Historical Summary Period

An ArcestrA graphic custom property can show historical statistics from Historian data over a defined summary period. During design time, you must specify the summary period to collect Historian summary data by entering values for the **Duration** and **StartTime** options shown on the **Edit Custom Properties** dialog box.

These assigned values are passed as input parameters of the new custom property. A value in minutes must be assigned to the **Duration** option.

The **StartTime** option can be left blank. Auto refresh is applied if a **StartTime** value is not specified.

- If a start time is not specified, then the start and end times of the summary period are calculated as:

Start Time = Current Time - Duration

End Time = Current Time

- If a start time is specified, then the start and end times of the summary period are calculated as:

Start Time = StartTime option value

End Time = StartTime + Duration

The **Duration** option can accept a negative number when a start time is specified. When **Duration** is assigned a negative number, the start time input parameter value becomes the end time of the summary period. The start time is calculated using the formula shown below:

End Time = StartTime assigned option value

Start Time = End Time + Duration (in this case it is negative value)

Duration can accept values from 1 minute to 10080 minutes, which is one week. **StartTime** must be within datetime Min and Max Value. During run time, history summary data is auto refreshed at an interval that is 25 percent of its duration length when a **StartTime** value is not specified.

Showing Statistical Summary Data

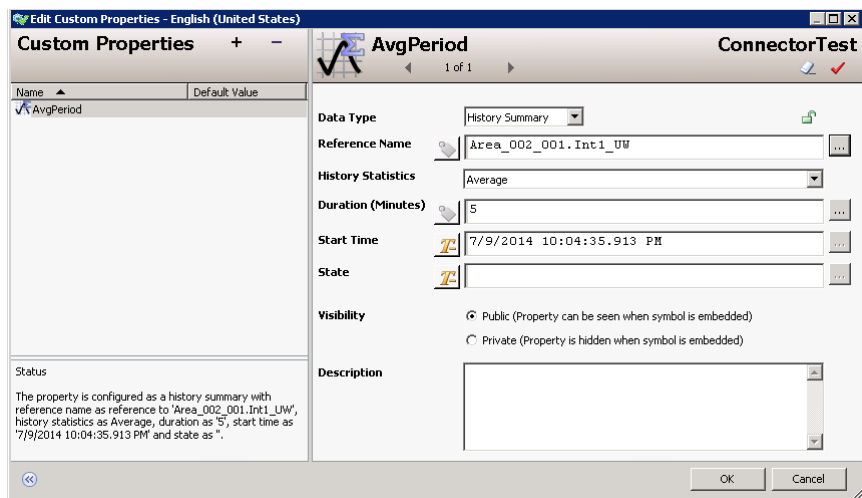
The following procedure explains how to specify custom property options to subscribe to historical statistical data. Before completing this procedure, you must have object or other data that is saved to the Historian and enabled for statistics.

To show historical summary data using custom properties

- 1 Add a custom property to an ArcestraA symbol.
 - a Click the canvas to cancel any selected elements.
 - b On the **Special** menu, click **Custom Properties**. The **Edit Custom Properties** dialog box appears.
 - c Click the **Add** icon. A new line is added in the custom properties list.
 - d Type a name for the new historical summary custom property and click **Enter**.
 - e You can see the names of the symbol and custom property in the header of the dialog box.
- 2 Select **HistorySummary** from the **Data Type** field.

Important: The History Summary data type only works with ArcestraA object attributes intended for InTouch managed applications. Do not attempt to use custom properties assigned the History Summary data type in Modern applications.

The **Edit Custom Properties** dialog box updates to show fields specific to the **HistorySummary** data type.



- 3 Enter a reference to data saved in the Historian in the **Reference Name** field.

The icon to the left of the **Reference Name** field toggles input to the field as **Static Text** or **Expression or Reference** mode.

Auto-Detect

The Historian server is auto-detected from the AppEngine on which the reference attribute is running. For example, if the **Reference Name** field is set to UDO.UDA1, the reference is set to the Historian server name configured for the AppEngine on which UDO is running.

Expression

When an expression or reference is typed in the **Reference Name** field, a connection is made to the specified Historian Server. The reference can be an external reference like an object attribute or a custom property.

Note: A reference cannot be made to historical data from an InTouch tag.

- 4 Select the type of historical statistics by selecting an option from the drop-down list of **History Statistics**.

Average is the default type of historical statistic. The following table shows the historical statistics options for analog and state summary data.

| Historical Statistics | Analog Historical Data | State Historical Data |
|------------------------------|-------------------------------|------------------------------|
| Average | ● | ● |
| First | ● | |
| Minimum | ● | ● |
| Maximum | ● | ● |
| Count | ● | ● |
| StdDev | ● | |
| Integral | ● | |
| PercentGood | ● | |
| Percent | | ● |
| Total | | ● |
| Last | ● | |

- 5 Set the length of the summary historical period in minutes by entering a value in the **Duration** field.

Acceptable **Duration** values are from 1 to 10080 and the default is 5. Duration can be specified as an integer, an expression, or a reference.

- 6 Set the start time of the of the summary period in the **StartTime** field.

A start time can be specified as static text, an expression, or a reference. The default start time is the current time.

A time for **StartTime** is optional and can be left blank. Auto refresh is applied if a **StartTime** value is not specified.

For more information about setting a start time, see "Historical Summary Period" on page 262.

- 7 Set the type of Historian summary data in the **State** field.

The **State** value can be expressed as an integer constant, static text, an expression, or a reference.

If a string value is provided, then string state summary data is queried from the Historian. If an integer value is entered, the Historian query is for analog state summary data.

State can be left empty. If empty, the default query is for analog summary data.

To get summary historical data for a Null state, enter "NULL" in the **State** field. The query checks for OpcQuality equal to openull and StringValue "NULL" in the result.

Using Binding in Custom Properties

ArchestrA object scripting supports a type called "Indirect". It enables you to bind a variable to a reference and read and write to it. This is done using the BindTo() method.

Note: The BindTo() method binds a variable to a reference as long as the symbol is shown.

For example, the local script variable ptr is defined and bound to the reference ud1_001.Int1.

```
dim ptr as indirect;
ptr.BindTo("ud1_001.Int1");
```

Within the same script you can use the indirect variable pointer to read from and write to the attribute ud1_001.Int1.

ArchestrA Symbols also use scripting in the same way as the scripting of Application Server.

However, as an ArchestrA Symbol can be embedded into an InTouch window and run anonymously, the time it takes to connect to the reference can be longer than one scan cycle.

For that reason, you cannot use the indirect variable immediately after it is bound to a reference to read from and write to it.

```
dim ptr as indirect;
```

```
ptr.BindTo("ud1_001.Int1");  
ptr = 42;
```

In the example, the value 42 cannot be written to the reference `ud1_001.Int1` as the binding takes longer.

To avoid this problem, you can modify your ArcestrA Symbol script to write the value after it is ensured that the binding is complete. The completion of the binding is indicated by the quality of the indirect variable.

You can configure a loop in the script to query for the quality and use the indirect variable to read from and write to the reference when its quality is good.

Note: Make sure to include an exit condition in your script, so that the script doesn't "hang" in case the binding cannot be made.

The following example script shows you how to do this:

```
dim ptr as indirect;  
  
dim timeout;  
  
ptr.BindTo("ud1_001.Int1");  
  
while (IsGood(ptr)==0); {if quality not good}  
    timeout=timeout+1; {increase the timer}  
    if timeout>10000 then {if timer reaches threshold}  
        exit while; {continue script execution}  
    endif;  
  
endwhile; {otherwise just loop for a while}  
ptr=42; {try to write to value to the reference}
```

A while loop is included in the script before the first write attempt. The while loop provides additional time for the symbol to connect to the reference. If the quality is good, then the script exits from the while loop.

Note: Similar behavior can occur when you try to bind to a reference of an object that is hosted on a different AppEngine.

Changing the Expression or Reference of a Custom Property at Run Time

You can change the expression or reference of a custom property at run time by calling the `SetCustomPropertyValue()` method on the symbol using a client script:

```
SetCustomPropertyValue(System.String name, System.String value,  
    System.Boolean isConstant);
```

You can select this method using the Element Browser from within the Symbol Editor. This method is supported only for ArcestrA client scripts.

This method has three parameters:

- *name* - Name of the custom property to be modified on the symbol. This parameter is of type string, and it can be a reference or a constant.
- *value* - The new value to be set. This parameter is of type string, and it can be an expression, reference, or constant. If the value is given in quotes ("), then the value is considered a constant. If the value is given without quotes, then the value of the expression is considered a reference.
- *isConstant* - A flag that indicates whether the new value will be evaluated as a constant or a reference. This parameter is of type Boolean. If it is set to True (1), then the new value will be treated as a constant. If it is set to False (0), then the new value will be treated as a reference. This parameter only applies when the *value* parameter is a reference or constant and the custom property specified in the *name* parameter is a string or time type. This parameter has no meaning if the custom property is an integer, float, Boolean, or double type.

Note: The *isConstant* parameter does not override the type of input for the *value* parameter. The *value* parameter itself can be either a constant or a reference depending on whether it is enclosed in quotes. The *isConstant* parameter is only determining how the actual value (coming from the *value* parameter) is evaluated.

The whole expression or reference of the custom property is replaced with the new value, regardless if it is overridden or not. No partial replacement is supported.

Only public custom properties on the symbol can be changed.

When the method executes, it overrides any modifications done by previous `IOSetRemoteReference()` calls from a native InTouch script.

For an example of configuring the custom property as a reference, say you have a `Motor_001` object with the string field attribute name `State` that stores the current state of the motor ("Running" or "Stopped"). You also have an `ArchestrA` symbol that has the string data type custom property `MotorState`. The following script code will set the `MotorState` custom property to `Motor_001.State` in run-time:

```
GraphicA.SetCustomPropertyValue("MotorState", "Motor_001.State",  
    False);
```

As a result of the call, the function will set the string custom property `GraphicA.MotorState` to `"Motor_001.State"` as a reference. The string custom property `GraphicA.MotorStatus` will resolve that reference and update its value with the reference value ("Running" or "Stopped").

For an example of configuring the custom property as a constant, say you have a `Motor_001` object with the Boolean field attribute `State` that reflects the current state of the motor (True or False). You also have an `ArchestrA` symbol that has the string data type custom property `MotorState`. The following script code causes the `MotorState` custom property to hold the state of equipment—"Running" or "Stopped"—as text based on the value returned for `Motor_001`:

```
IF Motor_001.State THEN  
  
    GraphicA.SetCustomPropertyValue("MotorState", "Running", True)  
    ;  
ELSE  
  
    GraphicA.SetCustomPropertyValue("MotorState", "Stopped", True)  
    ;  
ENDIF;
```

As a result of the call, the function will set the string custom property `GraphicA.MotorState` to "Running" or "Stopped," depending on the value of `Motor_001.State`.

Chapter 10

Animating Graphic Elements

You can use animations to change the appearance of graphic elements at run time. Animations are driven by data that comes from ArchestrA attribute values and expressions as well as element properties that can be changed in WindowMaker.

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 symbol or hide symbol.
- **Element-specific animations** for the Status element and Windows common control elements.

Each element in your ArchestrA Symbol 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 ArcestrA Symbol.

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 ArcestrA Symbol 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 270.

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 ArcestrA Symbol 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 ArcestrA Symbol 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 ArcestrA Symbol 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 ArcestrA Symbol 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.

Connecting Animations with Data Sources

You can connect animations to

- ArcestrA attributes
- Element properties
- Custom properties
- InTouch tags

For some input boxes, you can specify if the configuration is a static value or a reference by setting the input mode.

Connecting Animations with ArcestrA Attributes

You connect the element animation and appearance with an ArcestrA attribute. The ArcestrA attribute provides values at run time that control the behavior and appearance of the element.

For example, a rectangle element fill animation can be connected to the run-time value of the ArcestrA attribute Tank_001.PV.

You can browse all ArcestrA attributes, Element Properties, and InTouch tags with the Galaxy Browser.

You can select InTouch tags directly from the Galaxy Browser when working within the animation editor or the script editor. For more information, see "Using the Galaxy Browser InTouch Tag Browser Tab" on page 278.

To connect animations to attribute references using the Galaxy Browser

- 1 Select the element.
- 2 On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
- 3 Select the animation from the Animation list.
- 4 Select the parameter.
- 5 Click the browse button. The **Galaxy Browser** appears.
- 6 Click the **Attribute Browser** tab.
- 7 Select an AutomationObject from the list shown in the left pane. The attributes associated with the selected AutomationObject are shown in the right pane.
- 8 Select an attribute from the right pane and click **OK**. The selected attribute reference appears in the configuration box.
- 9 Repeat for any other animation parameters. Click **OK** when you are done.

Connecting Animations with Element Properties

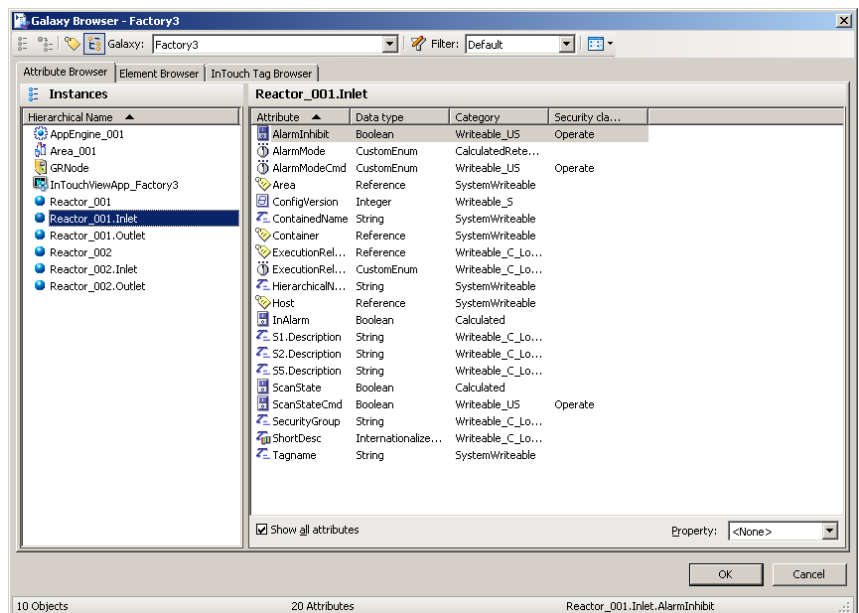
You can connect the element animation and appearance with a property of any element on the canvas.

You can browse the properties of all elements on the canvas with the Galaxy Browser.

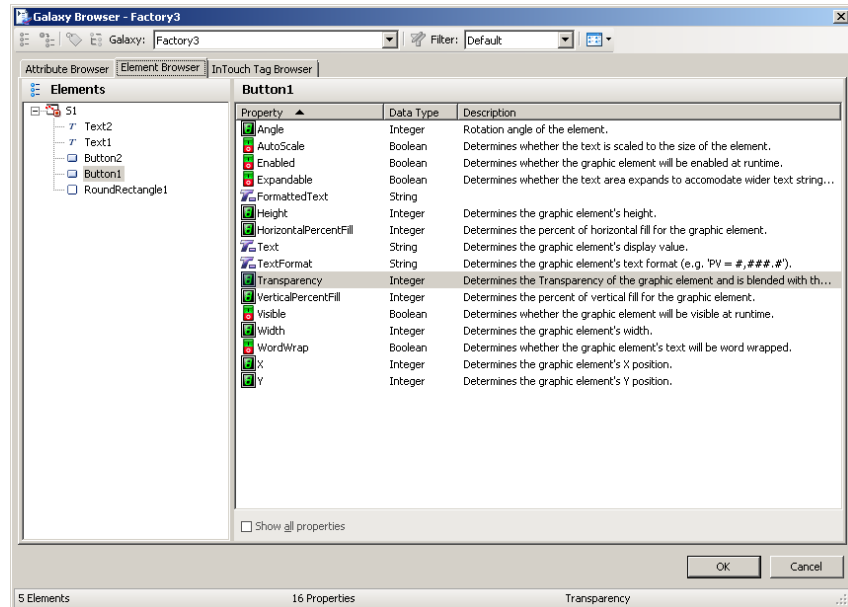
You cannot connect animations to properties of elements that are part of an embedded symbol on the canvas. You can connect animations to the public custom properties of embedded symbols.

To connect animations to element property references using the Galaxy Browser

- 1 Select the element.
- 2 On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
- 3 Select the animation from the Animation list.
- 4 Select the parameter.
- 5 Click the browse button. The **Galaxy Browser** appears.



- Click the **Element Browser** tab to show the **Element Browser** page.



- From the Elements List, select an element. The right pane shows the properties of the selected element.
- Select a property and click **OK**. The selected element and property appears in the configuration box.

Connecting Animations with Custom Properties

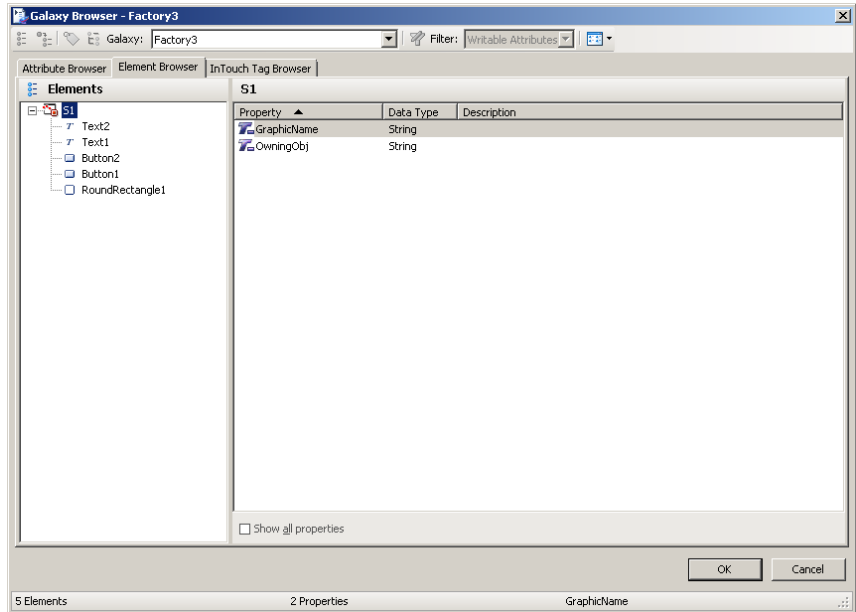
You can connect the element animation and appearance with a custom property of:

- The current symbol
- An embedded symbol on the canvas

To connect animations to custom property references using the Galaxy Browser

- Select the element.
- On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
- Select the animation from the Animation list.
- Select the parameter.
- Click the browse button. The **Galaxy Browser** appears.

- Click the **Element Browser** tab. The **Element Browser** tab appears.



- From the Elements List on the left, select the symbol. The right pane shows the custom properties and other properties of the selected symbol.
- Select a custom property and click **OK**. The selected custom property appears in the configuration box.

Connecting Animations with InTouch Tags

You can connect an element animation and appearance to an InTouch tag. The InTouch tag provides values at run time that control the animation and appearance of the element.

You can connect an element animation to an InTouch tag by:

- Configuring a reference with the **intouch:tag** syntax.
- Using a custom property and configuring the custom property in the embedded ArchestrA Symbol in InTouch to reference an InTouch tag. For more information, see the *InTouch HMI and ArchestrA Integration Guide*.
- Configuring an ArchestrA attribute reference to the managed InTouchViewApp object that contains the InTouch tags as attributes. The InTouchViewApp object uses the functionality of an InTouchProxy object.
- Configuring an ArchestrA attribute reference to an InTouchProxy object that contains the InTouch tags as items. This is a special case of configuring an ArchestrA attribute reference.

Using the InTouch:Tagname Syntax

When you use the **intouch:tagname** syntax, the animation connects to the InTouch tag of the node the symbol is used. There are some restrictions on how you can use this syntax:

- Unlike in Application Server, you cannot use true and false as Boolean values. Use 1 and 0 instead.
- If you want to make a reference to an InTouch SuperTag, use the following syntax instead:

```
attribute("intouch:SuperTag\Member")
```

Connecting Animations with InTouchViewApp Attributes

To be able to browse for InTouch tags, you must first:

- Create a managed InTouch application by deriving an InTouchViewApp template and configuring it in WindowMaker.
- Derive an instance of the InTouchViewApp derived template.

The InTouch tags are represented by attributes of the InTouchViewApp object instance.

Using the Galaxy Browser InTouch Tag Browser Tab

You can select InTouch tags directly from the Galaxy Browser when configuring a reference requiring an InTouch tag for an ArcestrA symbol animation or client script.

When invoked from either the animation editor or the script editor, the Galaxy Browser displays an **InTouch Tag Browser** tab in line with the **Attribute Browser** and **Element Browser** tabs.

The **InTouch Tag Browser** tab lists all InTouchViewApp instances and templates for the current Galaxy in the left pane. The right pane displays the InTouch tags for the selected InTouchViewApp. The **DotFields:** list box will display the dotfields associated with the selected tag.

The **Dotfields** list box below the right pane enables you to specify dotfields for the selected tag.

The **InTouch Tag Browser** tab behaves as follows:

- The InTouch Tag Browser functionality is available only from the animation editor or the script editor.
- The Galaxy Browser reads InTouch tags from the Tagname Dictionary.

- The Tagname Dictionary component is installed and available to the Galaxy Browser whether or not InTouch HMI is installed.
- Tags are refreshed only when the Galaxy Browser is closed and reopened.
- All tags remain in memory until you close and reopen the Galaxy Browser.
- If the InTouchViewApp is checked out by the current user, then the Galaxy Browser reads the latest Tagname Dictionary content of that InTouchViewApp.
- If the InTouchViewApp is checked in and is accessed by any user, then the Galaxy Browser always reads the checked-in version of the Tagname Dictionary.
- If the InTouchViewApp is checked out by a user other than the current user, the Galaxy Browser reads the most recently checked in Tagname Dictionary of that InTouchViewApp.
- If you select an InTouchViewApp template, the output reference string syntax is <InTouch:selectedTag>. If you select an InTouchViewApp instance, the output reference string syntax is <SelectedInTouchViewAppInstance.selectedTag>.

To connect animations to InTouch tags

- 1 Select the element.
- 2 On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
- 3 Select the animation from the Animation list.
- 4 Select the parameter.
- 5 Click the browse button. The **Galaxy Browser** appears.
- 6 Click the **InTouch Tag Browser** tab to show the **InTouch Tag Browser** page.
- 7 Select the InTouchViewApp object that corresponds to the managed InTouch application. The right panel shows the InTouch tags.
- 8 Select a tag and click **OK**. The selected ArchestrA reference to an InTouch tag appears in the configuration box.

Setting the Input Mode

In some boxes you can enter a value or expression that uses static and/or references to attributes and element properties. Boxes that support both input methods have an **Input Mode** selection icon.

Select:

- **Static Mode input** icon to specify literal static value or expression such as 3.141 or "Test".
- **Reference Mode input** icon to specify a reference to an attribute or element property such as: Tank_001.PV.

Note: To use static string values with or without references in Reference mode, you can enclose them with double-quotes such as: "Description: "+Tank_001.Desc

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 ArchestrA Symbol 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 ArcestrA 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 176.
- 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 176.
- 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 176.
 - 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" on page 284.
 - 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 176.

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 ArcestrA 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 176.
- 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 176.
- 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" on page 288.
- 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 style 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 ArcestrA 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 176.
- 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 176.
- 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" on page 291.
- 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:

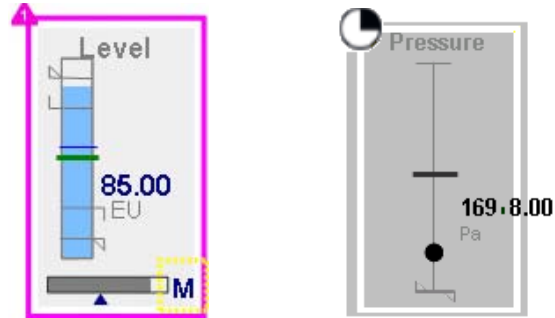
- The blinking speed: slow, medium or fast.
- If the element should blink invisibly or if it should blink with specified colors.

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 176.
- 9 Click **OK**.

Configuring an Alarm Border Animation

Alarm Border animation shows a highly visible border around a symbol or graphic element when an alarm occurs. The color and fill pattern of the border indicates the severity and current state of the alarm. 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 symbol:

- 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

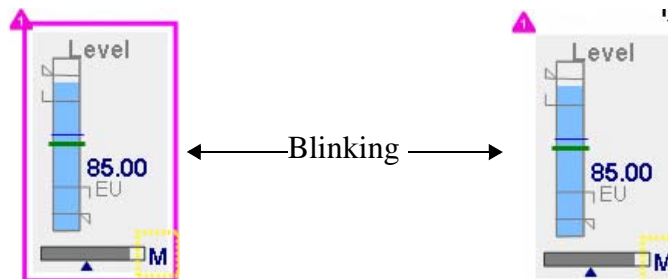
Alarm border animation can be applied to all types of symbols except embedded symbols and nested groups. Alarm border animation can also be applied to graphic elements and grouped elements.

Understanding the Behavior of Alarm Border Animations

Alarm Border animation appears around a graphic element based on the current state of an object's aggregate alarm attributes or an individual alarm's attributes. The appearance of the alarm border itself reflects alarm severity and alarm state.

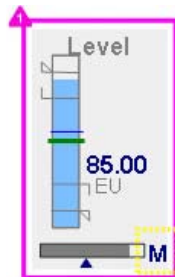
- A symbol's process value transitions into an alarm state.

Alarm Border animation appears around the symbol 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 symbol without blinking.



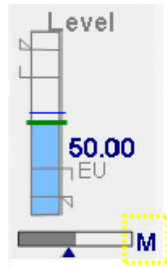
- The process value returns to normal without the user acknowledging the alarm.

Alarm Border animation remains around the symbol in a defined Return to Normal visual style without blinking.



- The alarm value returns to normal and the user acknowledges the alarm.

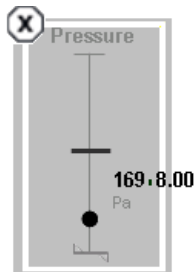
Alarm Border animation no longer appears around a symbol.



- The user suppresses an alarm

Alarm Border animation remains around the symbol in a defined suppressed/disabled visual style without blinking. The indicator shows the suppressed/disabled alarm mode icon.

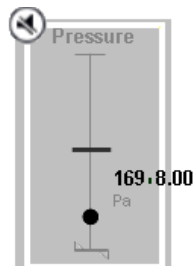
A suppressed alarm mode takes precedence over silenced and shelved modes.



- The user silences an alarm

Alarm Border animation remains around the symbol in a defined silenced visual style without blinking. The indicator shows the silenced alarm mode icon.

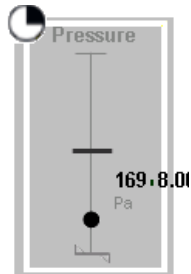
A silenced alarm mode takes precedence over an alarm in shelved mode.



- The user shelves an alarm

Alarm Border animation remains around the symbol in a defined shelved visual style without blinking. The indicator shows the shelved alarm mode icon.

Suppressed and silenced alarm modes take precedence over an alarm in shelved mode.



If a new alarm condition occurs when Alarm Border animation appears around a symbol, the animation updates to show the new alarm state. In the case of aggregate alarms, Alarm Border animation shows the highest current alarm state.

Understanding Alarm Border Animation Configuration

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 symbol.

The **Alarm Border** animation dialog box contains mutually exclusive fields to set the referenced attributes for aggregate or individual alarms.

For aggregate alarms, users specify Alarm Border animation by entering an attribute or object name in the **Use Standard Alarm-Urgency References** field of the **Alarm Border** dialog box.

The selected object attributes map to the following aggregate alarm attributes:

- AlarmMostUrgentAcked
- AlarmMostUrgentInAlarm
- AlarmMostUrgentMode
- AlarmMostUrgentSeverity
- AlarmMostUrgentShelved

For individual alarms, users 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, users 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. Based on the alarm state of these attributes, Alarm Border animation is applied to the graphic element in run time.

To configure Alarm Border animation

- 1 Open a symbol in the Symbol Editor.
- 2 Select the symbol to show the graphic elements listed in the **Elements** pane of the Symbol 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**.
Use Standard Alarm-Urgency References
 - a Click **Browse** and select an attribute or object name.
Both direct and relative references to an object are supported.
An expression cannot be used to reference the object.
 - b Click **OK**.**Use Customized Alarm-Urgency References**
 - a Click **Browse** and select an attribute, a symbol element, or an InTouch tag name for all **Source** fields shown beneath **Use Customized Alarm Urgency References**.
All fields must contain values and cannot be left blank.
Expressions, external references, and custom properties can be entered in all fields.
 - b Click **OK**.
- 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.

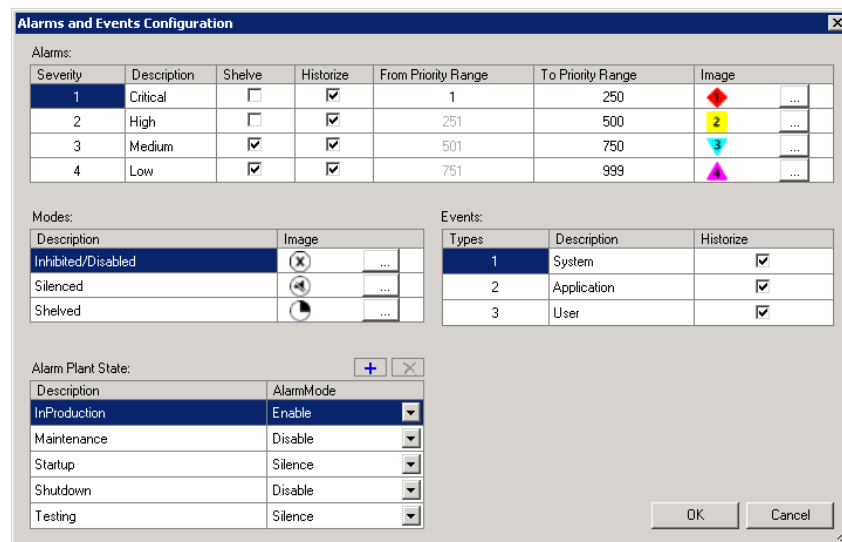
Configuring Optional Alarm Border Animation Characteristics

Users 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. 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 Events Configuration** dialog box. The images are saved in an XML file located in the ArcestrA global data cache.



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 **Alarms and Events Configuration**. The **Alarm and Events Configuration** 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 global data cache. 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. For more information about modifying Element Styles, see "Working with Element Styles" on page 193.

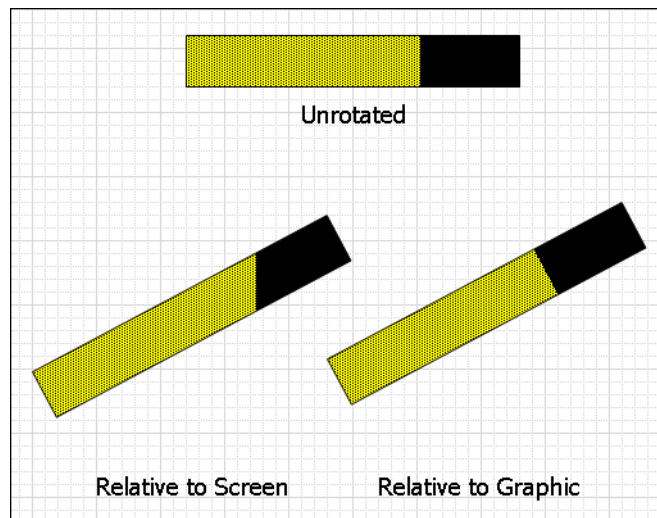
| Alarm Severity | Alarm State | Element Style |
|-----------------------|--------------------|----------------------------|
| 1 | UnAcknowledged | AlarmBorder_Critical_UNACK |
| 1 | Acknowledged | AlarmBorder_Critical_ACK |
| 1 | Return To Normal | AlarmBorder_Critical_RTN |
| 2 | UnAcknowledged | AlarmBorder_High_UNACK |
| 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 | Inhibited | AlarmBorder_Inhibited |
| All | Shelved | AlarmBorder_Shelved |
| All | Suppressed | AlarmBorder_Suppressed |
| 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 symbol 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:

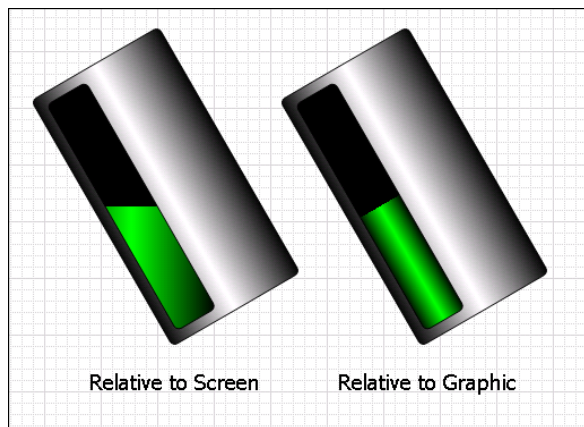
- In the **Analog** box, type an analog value, attribute reference or expression.
 - In the **Value - At Min Fill** box, type an analog value, attribute reference or expression that causes the minimum percent of filling at run time.
 - In the **Value - At Max Fill** box, type an analog value, attribute 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 176.
- 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 symbol 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 176.
- 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 symbol 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.

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.

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.

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 symbol elements, the animation point is the top left corner of the rectangle around all grouped elements.

To configure point animation

- 1 Open the symbol in Symbol 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 symbol. 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 symbol.
- 7 Enter an expression, constant, or reference in the Point field.
- 8 Repeat steps 6-7 to animate other points in the symbol.
- 9 Save your changes.

Configuring an Orientation Animation

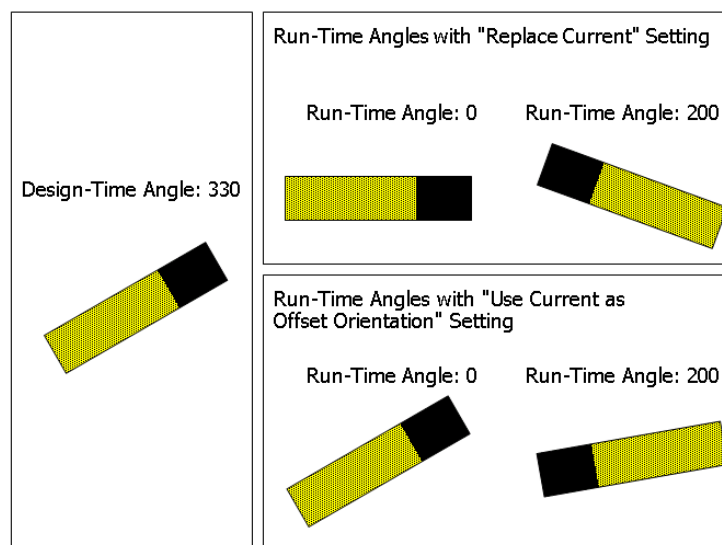
You can configure an element with an orientation animation. You can also:

- 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 page 320.
 - Selecting **Custom** from the **Text Format** list shows another drop-down list to select a number format.

Based on the selected **Custom** number format, the following fields appear on the **Analog Value Display** dialog box.

- **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.

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" on page 228.

Configuring Value Display Animation with the FormatStyle Property

When Value Display or User Input animation have a text format configured with a named global number style from the Galaxy Style library, 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 ArcestrA IDE or WindowViewer are 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 hours 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 name of the week day. |

| | |
|------|---|
| 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 01 to 12. |
| 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 symbol 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 the **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" on page 228.

Format String is the default numeric format, which includes a text entry field to assign a number format using four characters:

| Numeric Format Character | Description |
|---------------------------------|---|
| 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 managed InTouch applications. 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 without the century (2001 appears as "1"). |
| yy | The last two digits of the year (2014 appears as "14"). |
| yyyy | The full year (2014 appears as "2014"). |

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... |
|-------------------------------------|---|
| On Primary Click/Key Down | primary mouse button or a specific key is pressed. |
| While Primary Click/Key Down | primary mouse button or a specific key is pressed and held. |
| On Primary Click/Key Up | primary mouse button or a specific key is released |
| On Primary Double Click | primary mouse button is double-clicked. |
| On Secondary Down | secondary mouse button is pressed. |
| While Secondary Down | secondary mouse button is pressed and held. |
| On Secondary Up | secondary mouse button is released. |

| Use this... | To activate the action script when the... |
|----------------------------------|---|
| On Secondary Double Click | secondary mouse button is double-clicked. |
| On Center Down | center mouse button is pressed. |
| While Center Down | center mouse button is pressed and held. |
| On Center Up | center mouse button is released. |
| On Center Double Click | center mouse button is double-clicked. |
| On Mouse Over | pointer is moved over the element. |
| On Mouse Leave | pointer is moved out of the element. |
| On Startup | element is first shown in WindowViewer. |
| While Mouse Over | 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**.

Note: For more information about scripts, see "Adding and Maintaining Symbol Scripts" on page 369.

Configuring a Show Symbol Animation

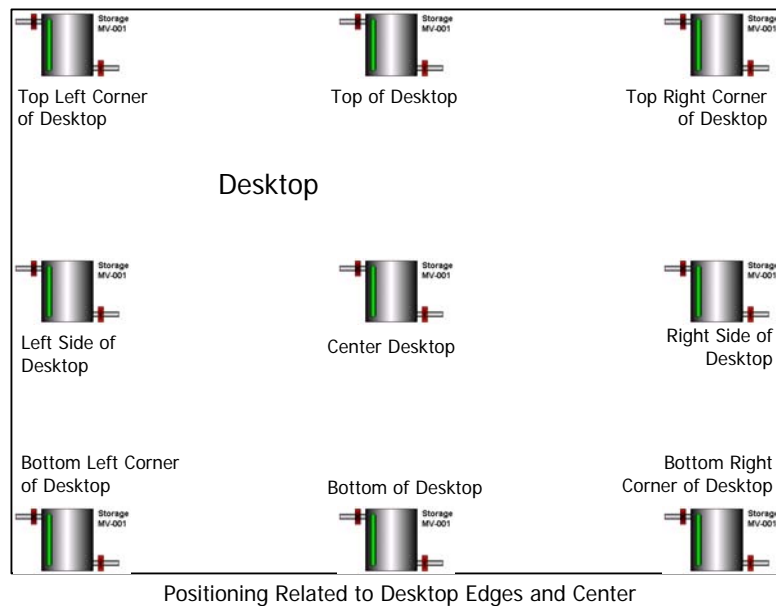
You can configure an element with a show symbol animation. A Show symbol animation shows a specified symbol in a new dialog box, when the element is clicked on.

You can configure:

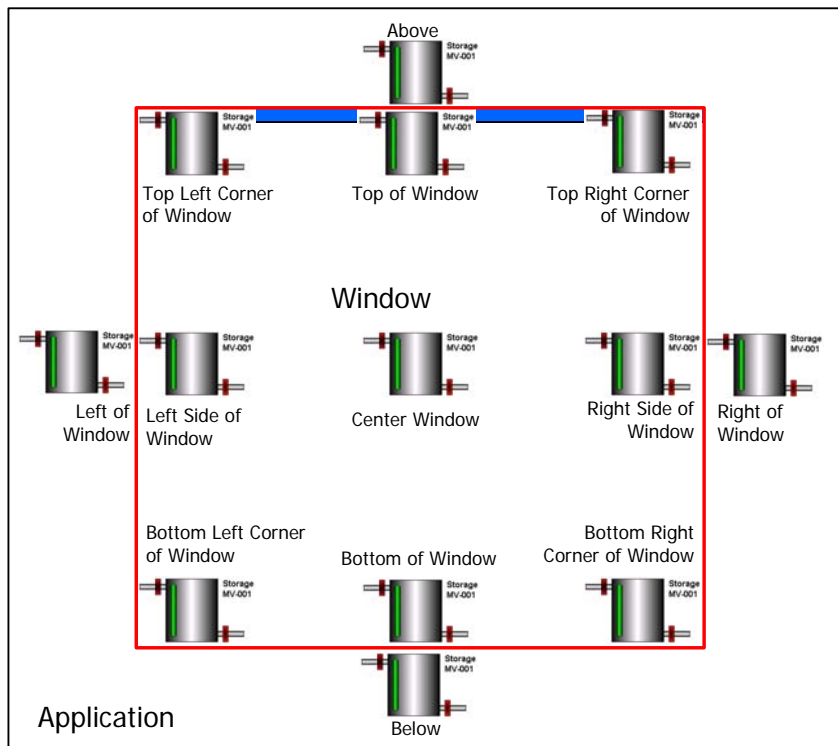
- Which symbol appears in the new window.
- If the window has a title bar, and if so if it has a caption.
- If the window is modal or modeless.
- If the window has a close button.
- If the window can be resized.
- The initial window position.
- The size of the window.

You can configure the position to be in relation of the:

- Desktop, such as at edges, corners, or at center.

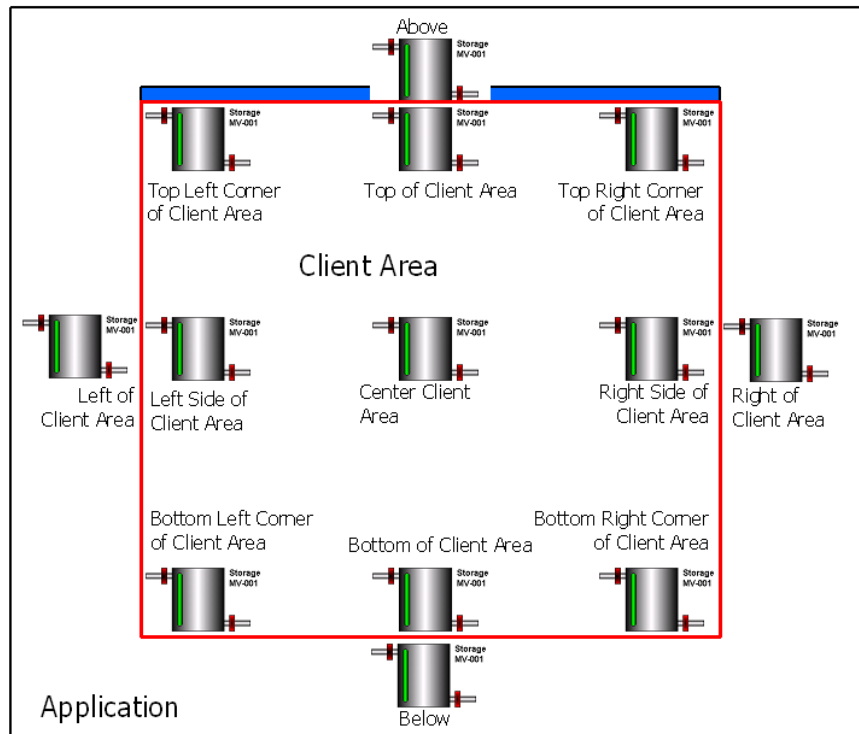


- Window, such as at one of its edges, its corners, its center or above, below, to the left or right. The window area includes the title bar if it appears.



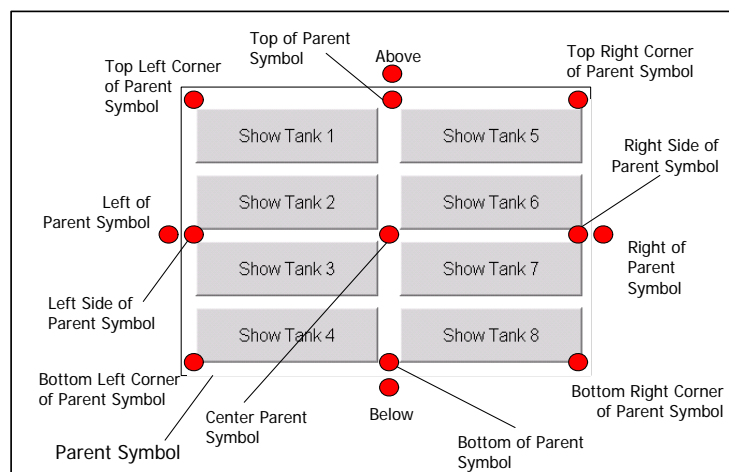
Positioning Related to Window

- Client Area. In InTouch client area is the active drawing area of an InTouch window excluding the title bar.



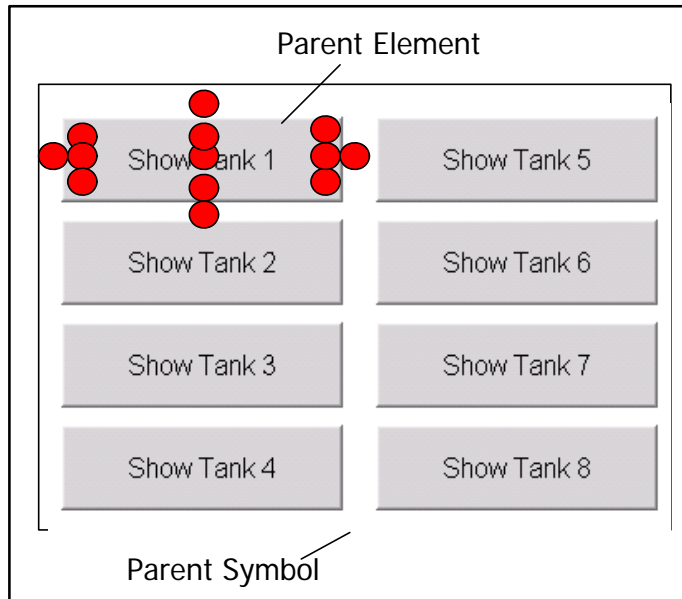
Positioning Related to Client Area

- 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.



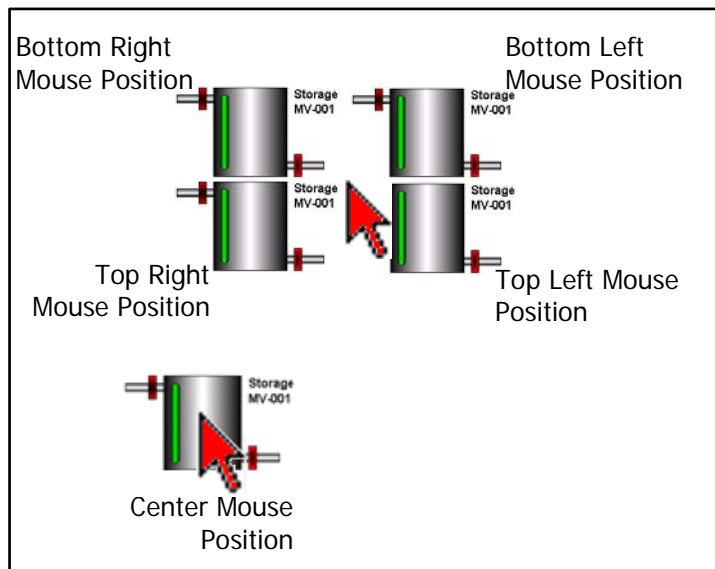
Positioning Related to Parent Symbol

- Parent Element, in which case the show symbol window is positioned in relation just to the element that called the show symbol window.



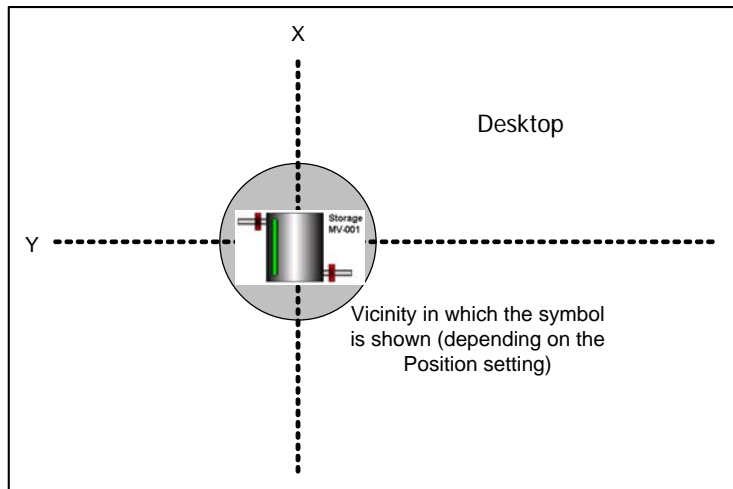
Positioning Related to Parent Element

- Mouse, in which case the show symbol window is positioned in relation to the pointer coordinates.



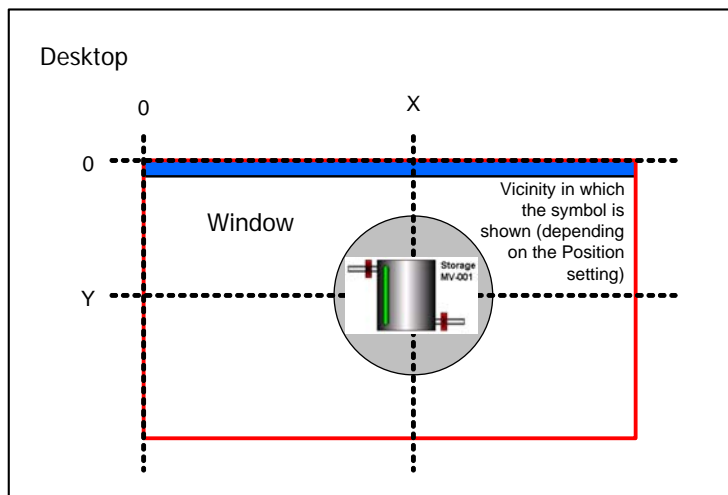
Selected Positionings Related to Mouse Pointer

- Desktop coordinates. The symbol is placed in the vicinity of coordinates that relate to the desktop.



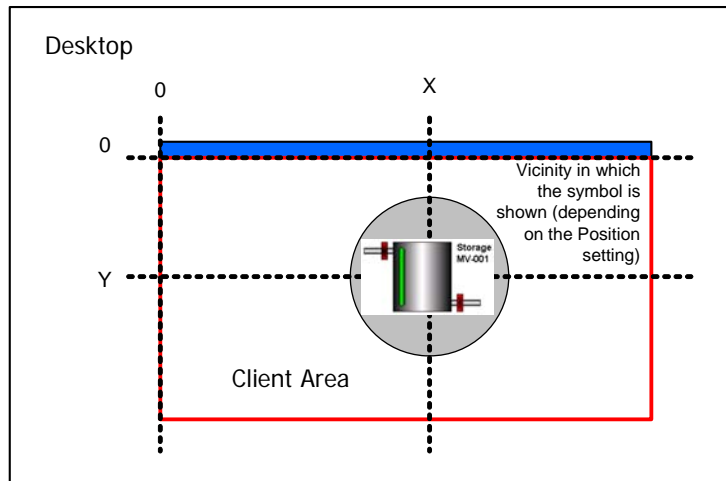
X, Y Positioning Related to Desktop

- Window coordinates. The symbol 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 symbol 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 symbol. Do one or more of the following:
 - a In the **Reference** box, type a symbol name or browse for one by using the browse button.
 - b To add a title bar to the symbol, select **Has Title Bar**.
 - c To use the symbol 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 symbol 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 symbol.
- **Custom Width and Height** to specify a width and height.

Depending on your selection of the item the symbol 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 symbol 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 symbol 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 symbol 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 symbol
- A symbol 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 symbol.
 - **Symbol shown by an element** if you want to close a symbol that appears by that element. Type the element name in the adjacent box.
- 5 You can specify that the symbol 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 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:

- ArcestrA attributes used in elements with animation.
- ArcestrA attributes 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 341.

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.

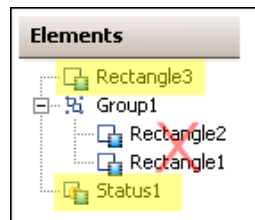
Tip You can also browse for the reference by clicking the browse button.

- 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 363.

- 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 363.
- 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 |

The user-defined attribute array **Options** of the **UD** object can appear as follows:

You can configure the array reference of the Radio Button Group as follows:

The Radio Button Group element appears as follows:

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 363.

- 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 363.
- 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 the Galaxy Browser with the Edit Box selected. For more information on these methods, see "Configuring Edit Box Methods" on page 382.

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 363.

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.

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 AutomationObject array.
- **Enum** combo box - uses captions and values contained in an enum data type of an AutomationObject.

You can also use Combo Box-specific methods in scripting to perform various functions at run time. You can browse these methods in the Galaxy Browser with the Combo Box selected.

For more information on these methods, see "Configuring Combo Box and List Box Methods" on page 383.

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 363.

- 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 363.

- 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 363.

- 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 363.

- 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 363.

- 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 `AutomationObject` array.
- **Enum** list box - uses captions and values contained in an enum data type of an `AutomationObject`.

You can also use List Box-specific methods in scripting to perform various functions at run time. You can browse these methods in the Galaxy Browser with the List Box selected.

For more information on these methods, see "Configuring Combo Box and List Box Methods" on page 383.

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 363.

- 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 363.

- 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 363.
- 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

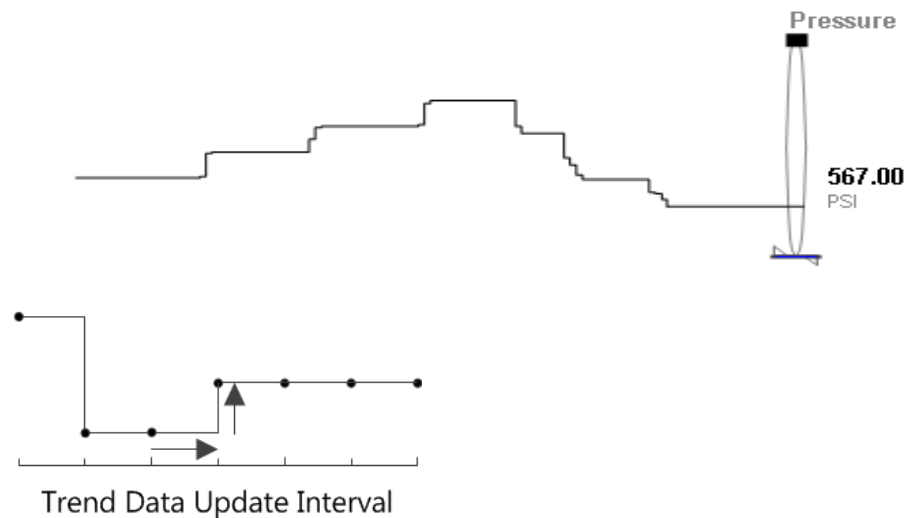
A Trend Pen shows a succession of process values as a trend line consisting of current and historical data updated at a minimum of one second intervals. A trend line gives operators a quick visual snapshot of a process value over a defined period.

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 Time 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.

You can configure two types of Trend Pen time periods.

- Moving time period

In a Moving trend period, the start time of a trend period is the current time. The end time is the duration of the time period from the start time.

$$\text{End Time} = \text{Start Time} - \text{Duration}$$

- 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 and 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. The `EndTime` property is read only.

$$\text{End Time} = \text{Start Time} + \text{Duration}$$

Understanding Trend Pen Historical Data Retrieval

When an application containing a Trend Pen starts running in WindowViewer, a Historian query retrieves data for the entire Trend Pen period. Real-time data is plotted on the trend line during the period that historical data is being retrieved. After retrieving historical data, it is added to real-time data to back fill the trend line for the entire period.

Important: A Trend Pen does not show historical data from an InTouch tag to prevent the possibility of showing invalid data when the Historian renames a tag on import. A Trend Pen only shows real-time data when a reference is made to an InTouch tag.

The following procedure shows the steps to configure a Trend Pen. Typically, configuring a Trend Pen includes several steps to place the Trend Pen next to a meter symbol to visually indicate the Trend Pen plot shows the changes in the symbol's process value over time.

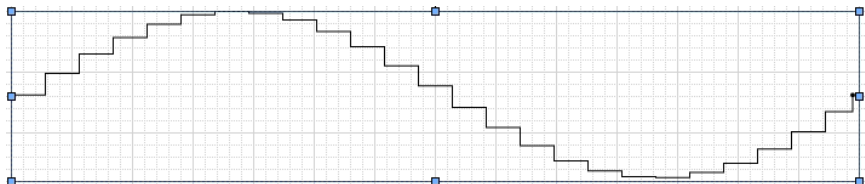
To configure Trend Pen

- 1 Click **Trend Pen** from the Graphic Toolbox.

The cursor changes to a cross hair when placed over the Symbol Editor canvas area.

- 2 Place the cursor over the canvas at the position you want to set as a horizontal boundary of the Trend Pen plot.
- 3 Pressing the left mouse button, drag the cursor to create a rectangle.

The horizontal and vertical boundaries of the Trend Pen graphic rectangle represent the drawing area of Trend Pen plot during run time. The horizontal axis of the graphic rectangle represents the time period of the trend. The vertical axis represents the range of the trend's possible values.



- 4 Release the left mouse key after sizing the rectangle to the height and width that you want to represent as the trend area.

The **Trend Pen** dialog box appears when the mouse key is released. You can also show the **Trend Pen** dialog box by double-clicking on the Trend Pen graphic or selecting **Edit Animations** from the **Special** menu.

- 5 Enter a reference in the **Reference** field.

The reference is the data source that appears as the value shown

by the trend, which can be an external reference like an object's attribute or a custom property. Constants and expressions are not allowed.

Note: A reference can be made to an InTouch tag, but the Trend Pen can only show real-time data.

- 6 Select **Auto-Detect** or **Expression** for the method to identify the location of the Historian.

Auto-Detect

The Historian server is auto-detected from the AppEngine on which the reference attribute is running. For example, if the **Reference** field is set to UDO.UDA1, then **Auto-Detect** is set to the Historian server name configured for the AppEngine on which UDO is running.

Expression

When an expression or reference is entered in the **Server Name** field, the Trend Pen connects to the specified Historian Server.

The icon to the left of the **Server Name** field toggles input to the field as an expression or Static Text mode.

A Trend Pen only shows live data if the **Server Name** field is left blank in **Expression** mode.

- 7 Select **Moving** or **Fixed** as the type of trend time period.

Moving

The start time of a trend period is the current time. The end time is the duration of the time period from the start time. The start time

for the next period is set to the end time of the previous trend period.

Fixed

In a Fixed trend time period, the `StartTime` and `EndTime` properties do not change automatically. The default start time of a trend period is the current time, but the `StartTime` property can be modified by a script to set a start time of a trend time period.

The `EndTime` property of a Fixed trend period is read only. The end time is calculated as the sum of the duration of the trend and the specified start time of the trend time period.

8 Set the X-axis time period of the trend line in the **Duration (Minutes)** field.

The trend time period can be specified as a constant, an external reference, an expression, or custom property. If a floating point number is entered, the period is rounded up to the nearest minute.

The minimum trend period is 1 minute and the maximum period is 10080 minutes (1week).

9 Select **Auto-Range** or **Clip out of Range Values** for the scaling method to place process values on a trend line.

If **Auto-Range** is selected, the **Min Range** and **Max Range** fields are disabled. The Y-axis of the trend line is automatically adjusted to show the full range of trend values within the upper and lower boundaries of the Trend Pen graphic.

If **Clip out of Range Values** is selected, the **Min Range** and **Max Range** fields are enabled. **Min Range** and **Max Range** set the lower and upper limits of the trend's Y-axis value range. Both fields can be set to constants, external references, or custom properties.

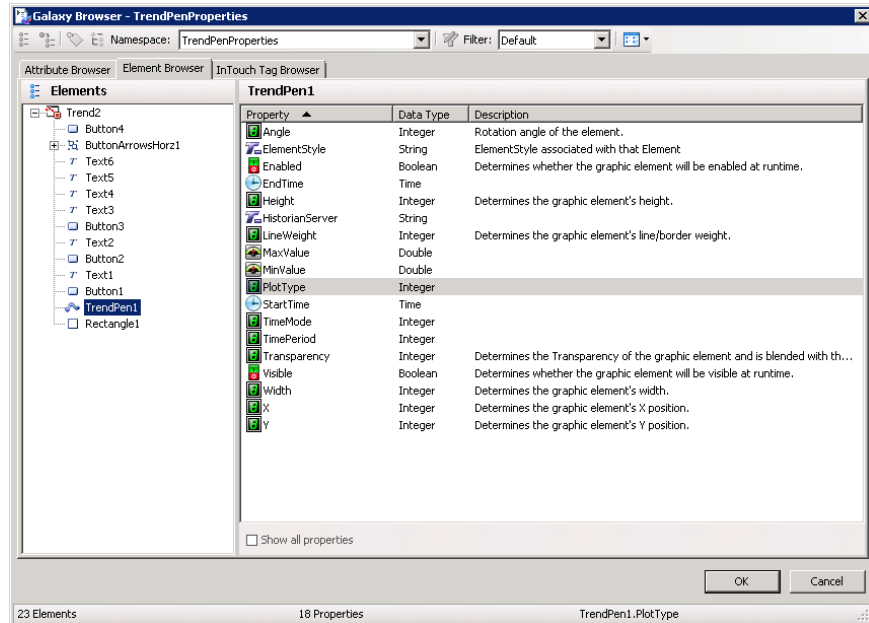
When a value exceeds the trend's minimum or maximum limits using **Clip out of Range Values**, the trend line is truncated at the limit of the value range and appears as a horizontal line for the period when the process value is out of trend's value range.

10 From **Plot Type**, select **Step Line** or **Line** as the type of trend plot.

- A **Step Line** plot draws a horizontal line from a trend data point to the time of the next data point on the trend's X-axis, and then draws a vertical line to the data point.
- A **Line** plot draws a line directly to each successive point within the trend period.

Changing Trend Pen Properties During Run Time

A Trend Pen contains a collection of properties whose values can be modified during run time to change the visual and functional characteristics of a trend.



The following properties are typically modified during run time to change the visual and functional characteristics of a Trend Pen.

MinValue Property

During run time, the value of the MinValue property can be modified to change the minimum measured value shown by a trend. During run time, MinValue can be a read/write or a read only property based on the value assigned to the Trend Pen's **Y Axis Range** option during design time.

- When **Y Axis Range** is set to **Auto-Range** during design time

The minimum measured value shown by a trend is set to the minimum value of data received from the Historian or from current data during the trend period. MinValue is read only.

- When **Y Axis Range** is set to **Clip out of range values** during design time

The minimum measured value shown by a trend is set to a minimum limit set from the **Min Range** option during design time.

MinValue is read/write and can be changed during run time. When MinValue is changed during run time, the trend line is redrawn based on the values assigned to the MinValue and MaxValue properties.

When the values assigned to MinValue and MaxValue properties are the same, the trend's Y Axis Range automatically changes to Auto-Range.

MaxValue Property

During run time, the value of the MaxValue property can be modified to change the maximum measured value shown by a trend. During run time, MaxValue can be a read/write or a read only property based on the value assigned to the Trend Pen's **Y Axis Range** option during design time.

- When **Y Axis Range** is set to **Auto-Range** during design time

The maximum measured value shown by a trend is set to the maximum value of data received from the Historian or from current data during the trend period. MaxValue is read only.

- When **Y Axis Range** is set to **Clip out of range values** during design time

The maximum measured value shown by a trend is set to a maximum limit set from the **Max Range** option during design time.

MaxValue is read/write and can be changed during run time. When MaxValue is changed during run time, the trend line is redrawn based on the values assigned to the MinValue and MaxValue properties.

When the values assigned to MinValue and MaxValue properties are the same, the trend's Y Axis Range automatically changes to Auto-Range.

StartTime Property

During run time, the value of the StartTime property can be modified to change the start time of a trend period based on the value set to the Trend Pen **Time Period** during design time.

- When **Time Period** is set to **Fixed** during design time

The default value assigned to the StartTime property is the start time set during design time. StartTime is read/write and can be changed during run time. When value of StartTime changes, the Trend Pen re plots the trend using new StartTime value.

- When **Time Period** is set to **Moving** during design time

The value set to the `StartTime` property is the current system date and time. `StartTime` is read only.

EndTime Property

During run time, the value of the `EndTime` property can be modified to change the end time of a trend period based on the value set to the Trend Pen **Time Period** during design time.

- When **Time Period** is set to **Fixed** during design time

The default value assigned to the `EndTime` property is the end time set during design time. `EndTime` is read/write and can be changed during run time. When value of `EndTime` changes, the Trend Pen re plots the trend using new `EndTime` value.

- When **Time Period** is set to **Moving** during design time

The value set to the `EndTime` property is the current system date and time. `EndTime` is read only.

PlotType Property

During run time, the value of the `PlotType` property can be modified to change the type of trend plot.

- When `PlotType` is 0, the Trend Pen plot type is Step Line. The default.
- When `PlotType` is 1, the Trend Pen plot type is Line.

The value of `PlotType` is ignored if the value is neither 0 nor 1.

When the value of `PlotType` changes in run time, trend data is retrieved again before drawing the trend.

TimeMode Property

During run time, the value of the `TimeMode` property can be modified to change the type of trend time period.

- When `TimeMode` is 0, the trend time period mode is Moving. The end time of the trend is the current time. The default.
- When `TimeMode` is 1, the trend time period mode is Fixed. The start time of the trend is the current time.

The value of `TimeMode` is ignored if the value is neither 0 nor 1.

When the time period changes from Moving to Fixed during run time, the trend's start time and end time remain the same before switching, and the data remains as well. When the time period changes from Fixed to Moving during run time, data is retrieved again before drawing the trend. The trend's start and end times are adjusted automatically by Moving mode.

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 for specifying the format of data during run time when the symbol is displayed. This allows you to change the style of the displayed data without changing the symbol 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 |

| Identifier | Type | Format | Example Output for "1.42" | Example Output for "-12400" |
|------------|----------------------------------|--------|---------------------------|-----------------------------|
| 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 |
| 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 "Format String Examples" on page 366). |

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 |

| Identifier | Type | Example |
|------------|--------------------------------|----------------------------------|
| T | Long time | 10:11:29 PM |
| f | Full date and time | December 10, 2002 10:11 PM |
| 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 GMT |
| 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 |

| Identifier | Type | Format | Example Output |
|------------|-----------|----------------|----------------|
| : | Separator | {0:hh:mm:ss} | 10:43:20 |
| / | 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 symbol by using the list

- 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 In the **New** column, type the reference to be replaced.
- 4 Click **OK**. All references are substituted accordingly in the elements.

To substitute references in a symbol 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 11

Adding and Maintaining Symbol Scripts

You can associate scripts to your symbols. Scripts can add animation to a symbol or its elements.

Caution: If you configure scripts that affect more than element and symbol animation, the script processing may affect performance.

About Symbol Scripts

You can configure your symbol with synchronous scripts that can be executed in run time.

You can:

- Configure the predefined scripts of a symbol.
- Add named scripts to a symbol.
- Edit existing named or predefined scripts in a symbol.
- Rename named scripts in a symbol.
- Remove named scripts from a symbol.
- Substitute references in named or predefined scripts.
- Use element methods in named or predefined scripts.

The autocomplete features available in the Application Server QuickScript editor are also available in the Graphic Editor script editor. For more information about Application Server scripting, see Working with QuickScript Editor Features in the *Application Server Scripting Guide*.

Predefined and Named Scripts

Predefined symbol scripts are similar to InTouch window scripts in that they run:

- One time when the symbol is shown or opened: **On Show**
- Periodically while the symbol is showing: **While Showing**
- One time when the symbol is hidden or closed: **On Hide**
- Any combination of the above

Named symbol scripts let you execute any number of scripts in run time that are triggered by values or expressions:

- 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 Symbol Scripts

When the symbol 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 InTouch tag whose quality is Initializing, or whose quality is Bad and category is not OK.

Security in Symbol Scripts

If the symbol script attempts to write to attributes 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 symbol script continues execution.

Signature Security for Acknowledging Alarms

SignedAlarmAck() is a script function for ArcestraA Graphics to perform an acknowledgment of one or more alarms on ArcestraA attributes 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 "SignedAlarmAck()" in Chapter 2, "QuickScript .NET Functions," in the *Application Server Scripting Guide*.

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.

Note: If the security mode is ArcestrA Galaxy security, then the domain displayed is "ArcestrA" and cannot 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" on page 91.

SignedAlarmAck() Scripting Tips

SignedAlarmAck() and Alarm Configuration

You can use the SignedAlarmAck() function only in ArcestrA client scripts.

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

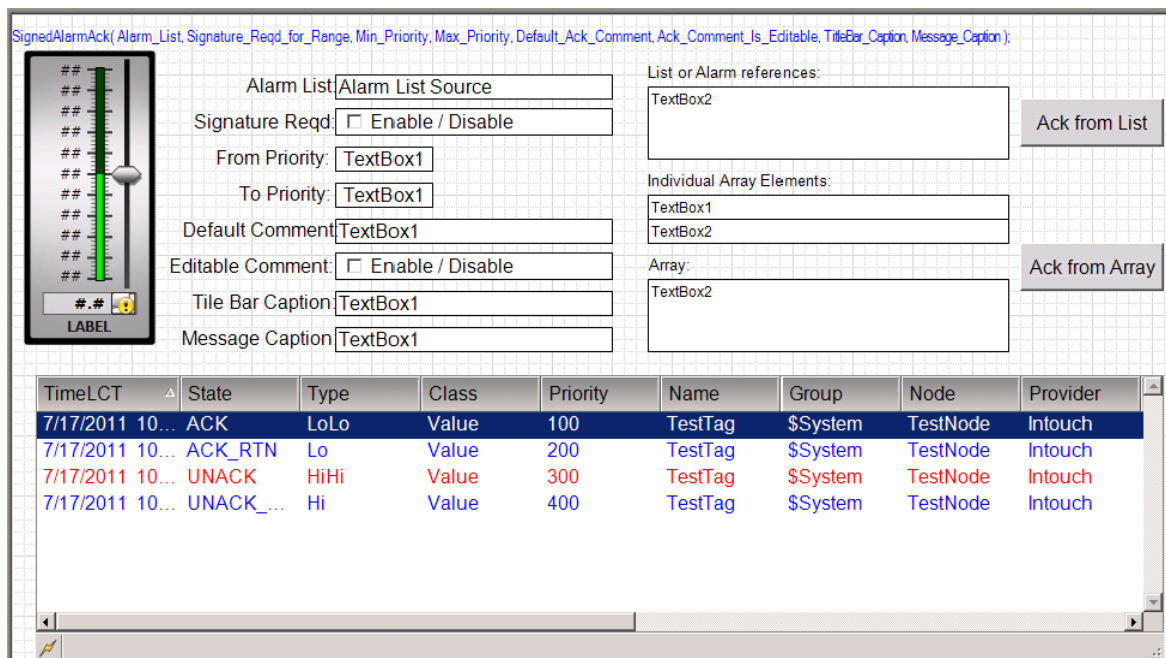
You can create a dashboard application to automate routine use of the SignedAlarmAck() function.

To configure signature required for alarm acknowledgement

- 1 Use the Graphic Editor to create a symbol. 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 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);
```

Symbol Script Time outs

To avoid infinite loops in a symbol 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 symbol 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**.

Error Handling

A symbol script does not run if it contains a syntax error. When the symbol or graphic is loaded, a message is written to the Logger.

Configuring the Predefined Scripts of a Symbol

You can configure the predefined scripts of a symbol. The predefined scripts can consist of:

- A script that runs one time when the symbol opens (On Show).
- A script that runs periodically as long as the symbol is open (While Showing).
- A script that runs one time when the symbol 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 symbol

- 1 Open the symbol in the ArcestrA Symbol 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 symbol opens.

If you create an OnShow script that uses a custom property bound to an InTouch tag, there is no guarantee that the tag data is valid when the script runs. This is because of the asynchronous nature of data subscriptions within ArcestrA. 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 377.

- **While Showing** to configure a script that runs periodically while the symbol is open.
 - **On Hide** to configure a script that runs one time when the symbol 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 symbol includes a custom property, the name of the custom property and a nested class property in the script cannot be the same.

6 Use the Script Function Browser and Attribute Browser to select external data.

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 InTouch 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 ArcestrA symbols, 1,000 ms.
- For migrated InTouch symbols, 0 ms; that is, the presence of external reference data is not checked.

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 symbols 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 Symbol

You can add named scripts to a symbol. 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 symbol

- 1 Open the symbol in the ArcestraA Symbol 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.

In this example, while the Boolean Controls.Filltank is true, the tank level Tan_001.pv increases by one unit every second.

Editing Symbol Scripts

You can edit predefined and named symbol scripts.

To edit symbol scripts

- 1 Open the symbol in the ArchestrA Symbol 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 symbol opens.
 - **While Showing** if the action script you want to edit runs periodically while the symbol is open.
 - **On Hide** if the action script you want to edit is runs one time when the symbol closes.
- 5 Edit the action script in the script box.
- 6 Click **OK**.

Renaming Scripts in a Symbol

You can rename named scripts in a symbol. When you rename the named script, the functionality of the script does not change.

To rename a named script

- 1 Open the symbol in the ArcestrA Symbol 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 press Enter. The script is renamed.

Removing Scripts from a Symbol

You can remove predefined or named scripts from a symbol.

To remove predefined scripts from a symbol

- 1 Open the symbol in the ArcestrA Symbol 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 symbol opens.
 - **While Showing** if the action script you want to remove runs periodically while the symbol is open.
 - **On Hide** if the action script you want to remove runs one time after the symbol closes.
- 5 Delete all the script content in the script box.
- 6 Click **OK**.

To remove named scripts from a symbol

- 1 Open the symbol in the ArcestrA Symbol 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 367.

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 symbol or for one of its elements, you can use the Galaxy Browser to show and select a:

- Property of an element
- Custom property of the symbol

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 symbol custom property

- 1 From the script window, click the Galaxy Browser icon. The **Galaxy Browser** dialog box appears.
- 2 Click the **Element Browser** tab. The Galaxy Browser shows the element names and the properties of the selected element.
- 3 Select an element or symbol from the list. The right pane shows the accessible properties of the selected element or symbol.
- 4 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 the Galaxy 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.

For more information on the methods, see "Overview of Windows Common Control List Methods" on page 555.

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.

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 hard drive 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 hard drive 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 12

Using Client Controls

You can:

- Import and embed client controls into a symbol.
- 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 symbols are using the client control.

For information on language switching for client controls, see Chapter 15, "Switching Languages for Graphic Elements."

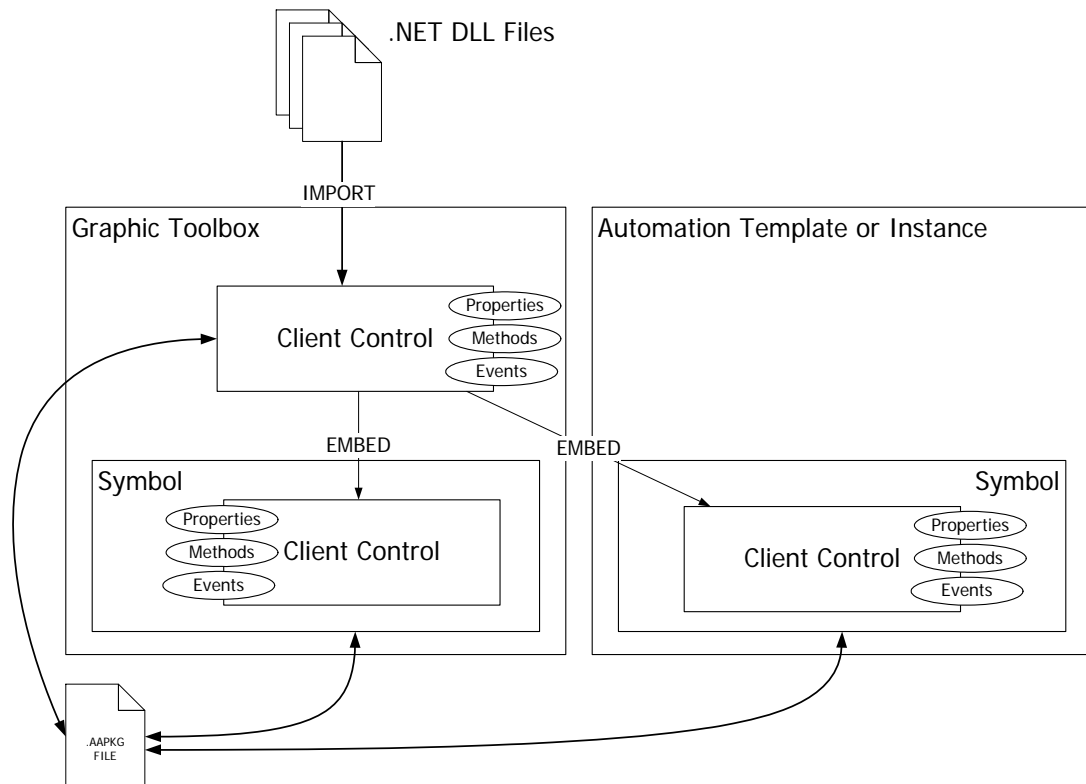
About Client Controls

Client controls give you functionality contained in .NET controls you can use in symbols. 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 symbol. The client controls appear as elements.
- View and edit the exposed client control properties.
- Bind the client control properties to ArchestrA attributes, symbol custom properties or InTouch tags. Do this using data binding animation.
- Configure scripts for client control. Do this using the animation.

You can then use the symbol containing the embedded client control in an InTouch application.

Wonderware ActiveFactory™ is one example of a software product that contains client controls.



Importing Client Controls

You can import client controls into the Graphic Toolbox from .NET Dynamic Link Library (.DLL) files.

After importing client controls, you can organize them in the Graphic Toolbox as you would with ArcestrA Symbols. For more information, see "Organizing Client Controls" on page 391.

You can also import client controls that have previously been exported in an ArcestrA package (.aaPKG) file.

If you import a newer version of a client control that you are already using in the ArcestrA IDE or in the InTouch HMI as embedded ArcestrA Symbol, you need to restart the ArcestrA IDE and/or the InTouch HMI.

Importing Client Controls

You can import one or more client controls from .DLL files. The client controls can be contained in one single .DLL file or span multiple files.

To import a client control, you must have security permissions to import graphic objects.

Note: If you select .NET .DLL files that do not contain client controls, the import process ignores these and continues at the next .DLL file.

To import client controls

- 1 In the IDE on the **File** menu, point to **Import**, and then click **Client Control**. The **Import Client Control(s)** dialog box appears.
- 2 Select one or more .NET .DLL files that contain the client controls you want to import and click **Open**. The **Import Preferences** dialog box appears.
- 3 Select the appropriate options for the import and click **OK**. The **Import Client Control(s)** dialog box appears.
- 4 When the client controls are imported, click **Close**. The imported client controls appear in the Graphic Toolbox.

Note: If the import fails, a message indicates the error in the **Import Client Control(s)** dialog box.

Example of Installing the ActiveFactory TagPicker Control

If you install Wonderware ActiveFactory 9.2 (or higher), you can install the client controls of one of the ActiveFactory *.DLL files. You can then use these controls to create an ArcestrA Symbol that contains the ActiveFactory TagPicker.

To install the ActiveFactory TagPicker client control

- 1 Open the IDE.
- 2 On the **File** menu, point to **Import**, and then click **Client Control**. The **Import Client Control(s)** dialog box appears.
- 3 Browse to the C:\Program Files\Common Files\ArcestrA folder, select the aaHistClientTagPicker.dll file and click **Open**. The **Import Preferences** dialog box appears.
- 4 Select the appropriate options for the import and click **OK**.
- 5 When the import is complete, click **Close**.
- 6 Open the **Graphic Toolbox** and expand the Galaxy node. aaTagPicker is listed as a client control.

Importing Previously Exported Client Controls

You can import one or more previously exported client controls from an ArcestrA package file (.aaPKG). Previously the client controls may have been:

- Exported without a symbol or an AutomationObject instance or template.
- Embedded in a symbol and the symbol was exported.
- Embedded in a symbol and contained in an AutomationObject instance or template and the AutomationObject was exported.

To import a previously exported ArcestrA package containing one or more client controls

- ◆ Import the exported client controls the same way as you would import an AutomationObject (.aaPKG). For more information, see the *Application Server User's Guide*.

Organizing Client Controls

You can organize the client controls within the Graphic Toolbox the same way as you would with ArcestrA Graphics. You can:

- Rename client controls.
- Move client controls in and out of Graphic Toolsets.
- Delete client controls.

For more information, see "Organizing Symbols in the Graphic Toolbox" on page 71 and "Importing and Exporting Symbols as ArcestrA Object Files" on page 74.

Embedding Client Controls

You can embed an installed client control into a symbol as you would embed a symbol within another symbol.

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 ArcestrA Symbol

- 1 On the **Edit** menu, click **Embed Graphic Symbol**. The **Galaxy Browser** appears.
- 2 Select the **Graphic Toolbox**.
- 3 Browse to the location that contains the client control.
- 4 Select a client control from the right panel and click **OK**. The pointer changes to paste mode.
- 5 Click on the canvas where you want to embed the client control. The client control is placed onto the canvas.

Example of Embedding the ActiveFactory TagPicker Client Control

If you install Wonderware ActiveFactory 9.2 (or higher), follow the steps of the "Example of Installing the ActiveFactory TagPicker Control" on page 390. Then do the following:

- 1 Create a new symbol in the Graphic Toolbox.
- 2 Open the symbol in the ArcestrA Symbol Editor.
- 3 On the **Edit** menu, click **Embed Graphic Symbol**.
- 4 Select **aaTagPicker** and click **OK**.

- 5 Click on the canvas near the top left corner. The ActiveFactory TagPicker control is placed on the canvas.

Viewing and Changing the Properties of Client Controls

When you embed a client control into a symbol, 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. For more information, see "Editing Common Properties of Elements and Symbols" on page 163.

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**.

Example of Changing a Property of the ActiveFactory TagPicker Control

Install and embed the ActiveFactory TagPicker Control into a symbol as described in:

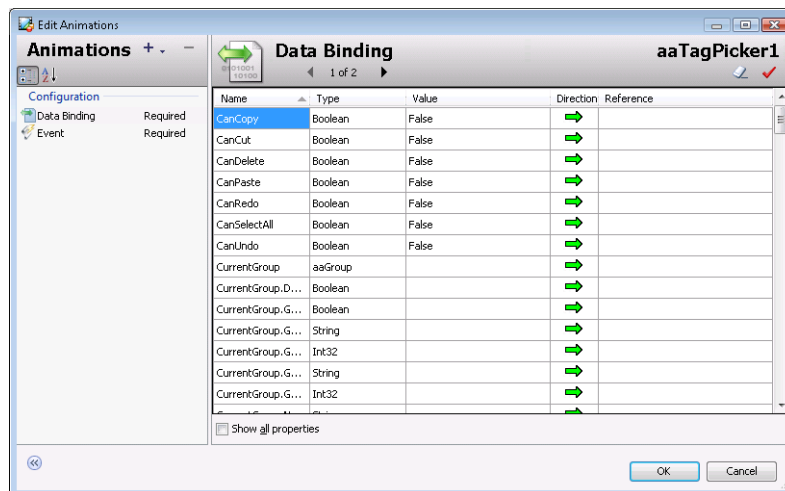
- "Example of Installing the ActiveFactory TagPicker Control" on page 390.
- "Example of Embedding the ActiveFactory TagPicker Client Control" on page 391.

In this example the "Tag Picker" caption of the TagPicker control is hidden.

- 1 Select the embedded ActiveFactory TagPicker client control. The Properties Editor shows all properties of the client control.
- 2 In the **Misc** property category, locate the property **HideCaption**.
- 3 Assign the value True to it and press Enter. The caption "Tag Picker" of the ActiveFactory TagPicker client control is hidden.

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**.

Example of Data Binding in the ActiveFactory TagPicker Control

Install and embed the ActiveFactory TagPicker control into a symbol as described in:

- "Example of Installing the ActiveFactory TagPicker Control" on page 390.
- "Example of Embedding the ActiveFactory TagPicker Client Control" on page 391.

In this example, the Boolean symbol custom property **HCV** controls the visibility of the ActiveFactory TagPicker caption.

Do the following:

- 1 Create a Boolean custom property and rename it **HCV**.
- 2 In the ArcestrA Symbol Editor, double-click the embedded ActiveFactory TagPicker control.
- 3 From the list of properties in the **Data Binding** configuration area, locate the **HideCaption** property.
- 4 Double-click the **Reference** box of the **HideCaption** property.
- 5 In the Galaxy Browser, select the **HCV** custom property and click **OK**.

The HideCaption property is now assigned to the element reference HCV.

- 6 Click **OK**.
- 7 Place a button on the canvas and configure it with a Boolean pushbutton animation that toggles the custom property HCV.
- 8 Save and close the symbol.
- 9 Embed the symbol in a managed InTouch application and test the data binding by clicking on the button in WindowViewer. When you do so, the visibility of the caption of the TagPicker control is toggled.

Configuring Client Control Event Scripts

You can configure an ArcestrA script that is executed when a client control event occurs. You do this using the **Event** animation.

To configure an ArcestrA 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.

Example of Configuring an Event Script for the ActiveFactory TagPicker Control

Install and embed the ActiveFactory TagPicker control into a symbol as described in:

- "Example of Installing the ActiveFactory TagPicker Control" on page 390.
- "Example of Embedding the ActiveFactory TagPicker Client Control" on page 391.

In this example, when one of the tags is picked by double-clicking on it, a message is logged in the Log Viewer.

First however, you need to:

- Import the script function library from the file aaHistClientDatabase.dll.
- Configure a connection to a valid and running Historian Server.

To import the script functions from aaHistClientDatabase.dll

- 1 On the **Galaxy** menu, point to **Import**, then click **Script Function Library**. The **Import Script Function Library** appears.
- 2 Browse to the aaHistClientDatabase.dll and select it. This file is by default in the C:\Program Files\Common Files\ArchestrA\ folder.
- 3 Click **Open**. The import starts and finishes with a message.
- 4 Click **OK**.

To connect the TagPicker control to the Historian Server

- 1 On the canvas, place a button next to the TagPicker control.
- 2 Double-click the button. The **Edit Animations** dialog box appears.
- 3 Add an **Action Scripts** animation to the animation list.
- 4 In the script area, type the following script:

```
Dim NewServer as ArchestrA.HistClient.Database.aaServer;  
Dim statusMessage as String;  
NewServer = aaTagPicker1.Servers.Add("MyHistorian");  
NewServer.LoginID = "MyUserName";  
NewServer.Password = "MyPassword";
```

```
NewServer.LogOn( statusMessage );
LogMessage ("Connection" + statusMessage);
```

5 In the script, replace the strings *MyHistorian*, *MyUserName* and *MyPassword* with the Historian server name, a valid user name, and a password to connect to the server.

6 Close the **Edit Animations** dialog box.

You can now configure the client control event to log a message every time the user picks one or more tags by double-clicking on them:

1 In the ArchestrA Symbol Editor, double-click on the embedded ActiveFactory TagPicker control.

2 In the animation list, click **Event**.

3 In the **Event** list, click the **OnTagsPicked** event.

4 In the script area, type the following:

```
LogMessage("User picked one or more tags.");
```

5 Save and close the **Edit Animations** dialog box.

6 Save and close the ArchestrA Symbol Editor.

7 Embed the symbol in a managed InTouch application.

8 Switch to run-time and connect to a valid IndustrialSQL Server source.

9 Double-click on one of the tags in the TagPicker control.

10 Check the SMC Log Viewer. The message "User picked one or more tags" appears.

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. For more information, see "Animating Graphic Elements" on page 269.

Exporting Client Controls

You can export client controls as ArcestrA package files (.aaPKG). You can export them:

- Directly from the Graphic Toolbox.
- Indirectly when you export AutomationObjects or symbols that reference them.

You can import the client controls again from the exported .aaPKG files.

To export client controls directly as ArcestrA package files

- 1 In the Graphic Toolbox, select one or more client controls that you want to export.
- 2 On the **Galaxy** menu, point to **Export**, and then click **Object(s)**.
- 3 Follow the general procedure for exporting AutomationObjects. For more information, see the *Application Server User's Guide*.

Securing Client Controls

The client controls use the same security setting as the symbols. You can set the security for client controls and symbols in the **Security** panel of the IDE. For more information, see "Configuring Security for Symbols" on page 87.

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>DynDepAsm1.dll</FileName>
    </DependentFile>
    <DependentFile>
      <FileName>DynDepAsm2.dll</FileName>
    </DependentFile>
    <DependentFile>
      <FileName>DynDepAsm3.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:element>
</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 cancelled.
- 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.

Viewing Additional Client Control Information

You can view:

- Which .DLL files, or assemblies, are used for the client control.
- The class name, vendor, and version.
- Which AutomationObjects and ArcestrA Symbols use the client control.

This information is contained in the Client Control Properties panels.

The client control properties are different than the properties of the embedded client control. The client control properties can be viewed in the IDE directly. The properties of the embedded client control can be viewed in the Properties Editor of the ArcestrA Symbol Editor.

Viewing the Client Control Assemblies

You can view which Client Control .DLL files, or assemblies, are used for the client control.

To view the client control assemblies

- 1 In the Graphic Toolbox, select the client control.
- 2 On the **Galaxy** menu, click **Properties**. The **Properties** dialog box appears.
- 3 On the **General** tab, you can view:
 - The **Primary Assembly**, which is the main .DLL file.
 - **Additional Assemblies**, which are linked to the main .DLL file and automatically loaded.

Viewing Class Name, Vendor, and Version of a Client Control

You can view the class name, vendor, and version of a client control in its **Properties** panel.

To view the class name, vendor, and version of a client control

- 1 In the Graphic Toolbox, select the client control.
- 2 On the **Galaxy** menu, click **Properties**. The **Properties** dialog box appears.
- 3 Click the **General** tab.

Viewing Objects and Symbols Referencing Client Controls

You can view which AutomationObjects and ArcestrA Symbols are using a given client control. This can be viewed in the **Properties** dialog box of the client control.

To view objects and symbols referencing a client control

- 1 In the Graphic Toolbox, select the client control.
- 2 On the **Galaxy** menu, click **Properties**. The **Properties** dialog box appears.
- 3 Click **Referenced By**. The list of objects and symbols using the client control is shown.

Chapter 13

Embedding Symbols within Symbols

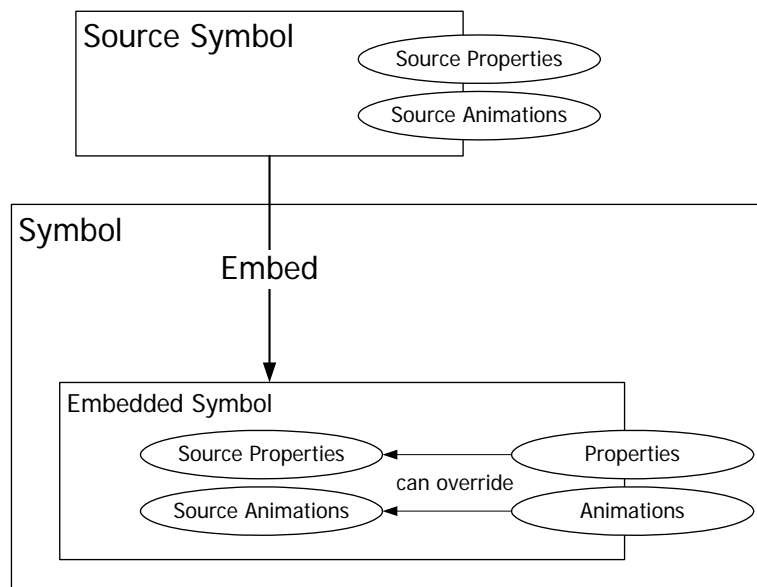
You can embed symbols into other symbols. Embedding symbols enables you to split your visualization into modules and reuse already defined symbols. For example, you can create a valve symbol and embed it multiple times into a tank symbol.

When you embed a symbol into another symbol, you are creating a link to the original symbol. Any changes to the original symbol are propagated to all embedded instances.

You can:

- Embed a symbol within another symbol.
- Edit an embedded symbol.
- Restore an embedded symbol to the original size of its source symbol.
- Convert the embedded symbol to a group.
- Detect the source of an embedded symbol.
- Edit the source of an embedded symbol.
- Override the custom properties of the source symbol.
- Control the size propagation of an embedded symbol.
- Select an alternate symbol or same symbol of an alternate AutomationObject instance as source.

- Edit the AutomationObject that contains the source symbol.
- Create a new instance of the AutomationObject that contains the source symbol.



Embedding Symbols

You can embed symbols from the Graphic Toolbox or an AutomationObject into other symbols.

When you embed a symbol, the animation links and scripts are inherited from the source symbol. You can only change the animations and scripts in the source symbol and all changes are propagated to the embedded symbol.

The embedded symbol appears with its original name appended by a number. The number is increased by one if you embed the same symbol again.

Note: If you embed symbols that have elements outside of the coordinates (0,0) and (2000,2000), the embedded symbol clips these elements.

Note: The name of the embedded symbol cannot be the same as a custom property of the symbol in which it is being embedded.

Note: The embedded symbol and the symbol in which it is being embedded cannot include elements that have the same name.

To embed source symbols from the Graphic Toolbox

- 1 On the ArcestrA Symbol Editor **Edit** menu, click **Embed Graphic Symbol**. The Galaxy Browser appears.
- 2 Select a source symbol from the Graphic Toolbox.
- 3 Click **OK**. The pointer appears in paste mode.
- 4 Click on the canvas to place the symbol.

To embed source symbols contained in an AutomationObject template

- 1 On the ArcestrA Symbol Editor **Edit** menu, click **Embed Graphic Symbol**. The Galaxy Browser appears.
- 2 Select the AutomationObject template that contains the source symbol.
- 3 Select the symbol 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 AutomationObject is created and the pointer appears in paste mode.
- 5 Click on the canvas to place the symbol.

To embed source symbols contained in an AutomationObject instance

- 1 On the ArcestrA Symbol Editor **Edit** menu, click **Embed Graphic Symbol**. The Galaxy Browser appears.
- 2 Select the AutomationObject instance that contains the source symbol.
- 3 Select the source symbol and click **OK**. The pointer appears in paste mode.
- 4 Click on the canvas to place the symbol.

Renaming Source Symbols and Hosting AutomationObjects

Generally, if you rename source symbols or their hosting AutomationObjects, embedded symbols update their references to the updated name of the renamed source symbol or hosting AutomationObject.

However, if you are using relative references and you rename the contained name of the referenced AutomationObject, the references to the embedded symbol are broken.

You can identify embedded symbols that may cause a problem by clicking on the embedded symbols 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 **AutomationObject** 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 Symbol

After you embed a source symbol into another symbol, its animations are inherited from the source symbol. The animation of the embedded symbol is controlled by the source symbol.

The embedded symbol itself has certain animations you can configure. The animations override the animations of the source symbol for the embedded symbol. 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 symbol to **True**:

- Tooltip
- User Input
- Slider Horizontal
- Slider Vertical
- Pushbutton
- Action Scripts

- Show Symbol
- Hide Symbol

To override the configured animations of an embedded symbol

- 1 Select the embedded symbol.
- 2 In the Properties Editor, change the value for the TreatAsIcon property to True.

Overriding Custom Properties of the Source Symbol

You can override the value and description of a custom property of the embedded symbol if the custom property's visibility is set to Public in the source symbol.

You cannot add, delete, or rename any custom properties of an embedded symbol 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 symbol.
- Set the visibility of the custom property. This has an effect if the symbol containing the embedded symbol is embedded into another symbol.

To override the value and description of a custom property

- 1 Select the embedded symbol 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 symbol 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 symbol.

Restoring an Embedded Symbol to the Original Size of its Source Symbol

You can restore an embedded symbol to its original size as it is defined in the object or Graphic Toolbox that contains it.

To restore an embedded symbol to its original size

- 1 Select the embedded symbol 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 symbol is restored to the original size of its source symbol.

Converting an Embedded Symbol to a Group

You can convert an embedded symbol to a group. A converted symbol is no longer associated with its source symbol. All configuration of the embedded symbol is preserved.

If you convert an embedded symbol to a group:

- Scripts of the embedded symbol are not converted.
- You can optionally move the custom properties to the group.
- Relative references of the embedded symbol are no longer valid.

To convert an embedded symbol to a group

- 1 Select the embedded symbol 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 symbol is converted to a group.

Detecting the Source Symbol of an Embedded Symbol

You can view the source of an embedded symbol by using the `SymbolReference` property.

To detect the source of an embedded symbol

- 1 Select the embedded symbol 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 symbol itself. This can be:
 - `Symbol:SymbolName`.
 - `Symbol:InstanceName.SymbolName`.

Editing the Source of an Embedded Symbol

You can edit the source of an embedded symbol by opening it in a new session of the ArcestrA Symbol Editor.

To edit the source of an embedded symbol

- 1 Select the embedded symbol.
- 2 On the **Edit** menu, point to **Embedded Symbol**, and then click **Edit Symbol**. The source of the embedded symbol is opened in a new session of the ArcestrA Symbol Editor.
- 3 Edit the source symbol as needed and click **Save and Close**. The new session of the ArcestrA Symbol 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 symbol.

If you do not accept the change, the embedded symbol is updated the next time you open it in the ArcestrA Symbol Editor.

Controlling Size Propagation of Embedded Symbols

You can control the way that size changes of the source symbol are propagated to its embedded instances, which are embedded symbols. For example, a size change is:

- Resizing one of the elements in the source symbol so that the symbol boundary changes.
- Adding elements to or removing elements from the source symbol so that the symbol's boundary changes.

This feature is called dynamic size change and can be enabled or disabled.

Setting the Anchor Point of a Source Symbol

You can set the position of the anchor point of a source symbol. The anchor point of a source symbol 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 symbol 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 Symbols

You can show or hide the anchor points of embedded symbols. An anchor point shows the current absolute anchor position of the embedded symbol on the canvas.

To show or hide the anchor point of an embedded symbol

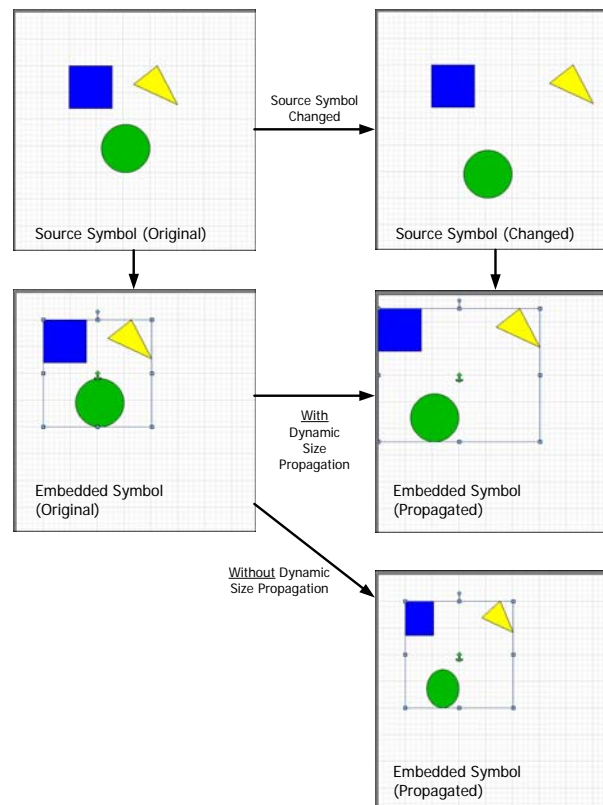
- ◆ On the toolbar, click the **Show/Hide Embedded Symbol Anchor Points** icon. The anchor of the embedded symbol appears or disappears.

Enabling or Disabling Dynamic Size Change of Embedded Symbols

You can enable or disable the dynamic size change of embedded symbols. The anchor points of the embedded instances are not changed by any size change to the source symbol.

If the source symbol size changes and the dynamic size change is enabled, the embedded symbol size adapts accordingly. If the dynamic size change is disabled, the embedded symbol 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 symbol

- 1 Select the embedded symbol 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 Symbols and Instances

If your embedded symbol is contained in an AutomationObject instance, you can:

- Use another symbol that is contained in the same AutomationObject instance.
- Use the same symbol that is contained in a different AutomationObject instance, in which case the animation links in the symbol are redirected.

Selecting Alternate Symbols

You can select an alternate symbol of the same AutomationObject instance to embed. The following properties are retained:

- Position and size
- Animations applied to the embedded symbol
- Override information (the TreatAsIcon property)

You can only select alternate symbols for embedded symbols contained in AutomationObject instances.

To select an alternate symbol

- 1 Select the embedded symbol on the canvas.
- 2 On the **Edit** menu, point to **Embedded Symbol**, and then click **Select Alternate Symbol**. The Galaxy Browser appears.
- 3 If available, select an alternate symbol that is contained in the same instance and click **OK**. The embedded symbol is updated with the new alternate symbol.

Selecting Alternate Instances

You can select an alternate instance of the AutomationObject that contains the same embedded symbol. When you select an alternate symbol to embed, the following properties are retained:

- Position and size
- Animations applied to the embedded symbol
- Override information (the TreatAsIcon property)

To select an alternate instance

- 1 Select the embedded symbol on the canvas.
- 2 On the **Edit** menu, point to **Embedded Symbol**, and then click **Select Alternate Instance**. The Galaxy Browser appears with a list of all instances that contain the same symbol.
- 3 Select an instance and click **OK**. All internal references of the embedded symbol update to point at the alternate instance. The name of the embedded symbol updates to reflect that it is pointing at a different instance.

Detecting and Editing the Containing AutomationObject Instance

You can detect and edit the AutomationObject instance that contains the embedded symbol.

To detect the AutomationObject instance that contains the source symbol

- 1 Select the embedded symbol.
- 2 In the Properties Editor, locate the OwningObject property. Its value contains the name of the object that contains the source symbol.

Note: You can write to this property at run time to force the embedded symbol to point to a different AutomationObject in its references contained in animation links.

To edit the AutomationObject that contains the source symbol

- 1 Select the embedded symbol.
- 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 AutomationObject

You can create a new instance of the AutomationObject that contains an embedded symbol. The following properties of the symbol are retained:

- Position and size
- Animations
- Override information (the TreatAsIcon property)

To create a new instance of the AutomationObject that contains an embedded symbol

- 1 Select the embedded symbol.
- 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 AutomationObject is created and the references and name of the embedded symbol are updated to point at it.

Chapter 14

Migrating InTouch SmartSymbols

You can use InTouch SmartSymbols in ArcestrA symbols by importing (migrating) them. The SmartSymbol appearance and configuration is imported and converted to animation configuration.

The imported SmartSymbol can:

- Be added to the existing elements on the canvas.
- Replace the existing elements on the canvas.

Importing InTouch SmartSymbols into an ArcestrA Symbol

Generally, you can import any InTouch SmartSymbol into an ArcestrA symbol.

Note: The SmartSymbol can contain objects that cannot be imported, or can be imported but have limited functionality. For a full list of these objects, see "Restrictions for SmartSymbol Import" on page 419.

To import an InTouch SmartSymbol into an ArcestrA Symbol

- 1 Open the ArcestrA Symbol Editor.
- 2 On the **Special** menu, click **Import InTouch SmartSymbol**. The **Find InTouch Application Wizard** appears.
- 3 If your InTouch application is in a different folder than the default, click the browse button to browse for the path to the InTouch application.
- 4 Select **Find Applications**. The **Search Root** box shows the path under which all applications are to be searched.
- 5 If your InTouch application is not under the specified **Search Root** path, change the **Search Root** path by typing a new start folder for the search or browsing for one.
- 6 Click **Find**. All InTouch applications contained in the specified **Search Root** folder are found and listed.
- 7 Select the application from which you want to import SmartSymbols and click **Next**. The **Select InTouch SmartSymbol** dialog box appears.
- 8 Browse to the location of the SmartSymbol in the SmartSymbol hierarchy, select the SmartSymbol that you want to import, and then click **OK**.
- 9 If you already have elements on the canvas, a dialog box appears prompting if you want to replace the existing elements. Click:
 - **Yes** if you want to delete the existing elements and import the SmartSymbol on an empty canvas.
 - **No** if you want to keep the existing elements and import the SmartSymbol.
- 10 If the SmartSymbol contains fonts that are currently not installed on the operating system, the **Edit Font Mapping** dialog box appears.

You can click **Continue** to accept the suggested font mapping, or change the font mapping for each individual font. To do this:

- a Click the font name in the **Mapped Font** column. A browse button appears.
- b Click the browse button. The **Supported Font Selection** dialog box appears.
- c Select a font from the list and click **OK**. The selected font appears in the **Mapped Font** column.
- d Repeat the steps for any other font you want to map to another font.

Note: If you want to save the mapping for the next time you import a SmartSymbol, check **Save mapping**.

11 The SmartSymbol is imported and appears on the canvas.

Restrictions for SmartSymbol Import

When you import an InTouch SmartSymbol, the following configuration is imported:

- InTouch graphics
- Graphical animations
- Scripts
- References

Importing InTouch Graphics

The following tables shows you InTouch graphics that:

- Can be imported without any problem.
- Can be imported but are changed in their functionality or lose some functionality in the process.
- Cannot be imported.

The following InTouch graphics can be imported without any problem:

| InTouch Graphic | ArchestrA Graphic Element | Notes |
|-------------------|---------------------------|---|
| Rectangle | Rectangle | |
| Rounded Rectangle | Rounded Rectangle | |
| Ellipse | Ellipse | |
| Line | Line | |
| H/V Line | Line | Smart Symbols convert H/V lines to Lines. Therefore, ArchestrA can only generate lines. |
| Polyline | Polyline | |
| Polygon | Polygon | |
| Text | Text | |
| Bitmap | Bitmap | |

| InTouch Graphic | ArchestrA Graphic Element | Notes |
|------------------------|----------------------------------|---|
| Cell | Group | ArchestrA property "Treat as Icon" = false. |
| Symbol | Group | ArchestrA property "Treat as Icon" = true. |
| Button | Button | |

The following InTouch graphics can be imported, but are changed in their functionality or lose some functionality in the process:

| InTouch Graphics | ArchestrA Graphic Element | Notes |
|-------------------------|----------------------------------|---|
| Wizard | Elements | When grouped in a SmartSymbol, it appears as a group of elements. |
| SmartSymbol | Elements | When grouped in another Smart Symbol, it is broken down into a cell, losing its SmartSymbol properties. |

The following InTouch graphics cannot be imported, as they cannot be added to a SmartSymbol:

| InTouch Graphic | ArchestrA Graphic Element | Notes |
|------------------------|----------------------------------|--|
| RealTime Trend | n/a | Cannot be added to a SmartSymbol. |
| Historical Trend | n/a | Cannot be added to a SmartSymbol. |
| ActiveX Controls | n/a | Cannot be added to SmartSymbol. This would include all ActiveX alarm controls (Alarm DB View, Alarm Viewer, and so on) |

Importing Graphical Animation

When you import an InTouch SmartSymbol, all data configured in InTouch animations is imported to ArcestrA animations. InTouch animations and ArcestrA animations often have a different name, but perform the same function.

The following table shows you which animations correspond to each other:

| InTouch Animation Link | ArcestrA animation |
|------------------------------------|---------------------------|
| User Inputs - Discrete | User Input |
| User Inputs - Analog | User Input |
| User Inputs - String | User Input |
| Sliders - Vertical Slider | Vertical Slider |
| Sliders - Horizontal Slider | Horizontal Slider |
| Touch Pushbuttons - Discrete Value | Pushbutton |
| Touch Pushbuttons - Action | Action Scripts |
| Touch Pushbuttons - Show Window | not supported |
| Touch Pushbuttons - Hide Window | not supported |
| Line Color - Discrete | Line Style |
| Line Color - Analog | Line Style |
| Line Color - Discrete Alarm | Line Style |
| Line Color - Analog Alarm | Line Style |
| Fill Color - Discrete | Fill Style |
| Fill Color - Analog | Fill Style |
| Fill Color - Discrete Alarm | Fill Style |
| Fill Color - Analog Alarm | Fill Style |
| Text Color - Discrete | Text Style |
| Text Color - Analog | Text Style |
| Text Color - Discrete Alarm | Text Style |
| Text Color - Analog Alarm | Text Style |
| Object Size - Height | Height |
| Object Size - Width | Width |
| Location - Vertical | Location Vertical |

| InTouch Animation Link | ArchestrA animation |
|--------------------------------|----------------------------|
| Location - Horizontal | Location Horizontal |
| Percent Fill - Vertical | % Fill Vertical |
| Percent Fill - Horizontal | % Fill Horizontal |
| Miscellaneous - Visibility | Visibility |
| Miscellaneous - Blink | Blink |
| Miscellaneous - Orientation | Rotation |
| Miscellaneous - Disable | Disable |
| Miscellaneous - Tooltip | Tooltip |
| Value Display - Discrete Value | Value Display |
| Value Display - Analog Value | Value Display |
| Value Display - String Value | Value Display |

Importing Action Scripts

When you import a SmartSymbol, all action scripts associated with objects in SmartSymbol are imported as well. An action script in a SmartSymbol becomes a script animation in an ArchestrA Symbol.

Most of the predefined InTouch functions (QuickScripts) are imported.

Mathematical Functions

The following mathematical functions in InTouch WindowMaker are supported by the ArchestrA Symbol Editor:

Abs, ArcCos, ArcSin, ArcTan, Cos, Exp, Int, Log, LogN, Pi, Round, Sgn, Sin, Sqrt, Tan, Trunc

String Functions

The following string functions in InTouch WindowMaker are supported by the ArchestrA Symbol Editor:

Dtext, StringASCII, StringChar, StringCompare, StringCompareNoCase, StringFromGMTTimeToLocal, StringFromIntg, StringFromReal, StringFromTime, StringFromTimeLocal, StringInString, StringLeft, StringLen, StringLower, StringMid, StringReplace, StringRight, StringSpace, StringTest, StringToIntg, StringToReal, StringTrim, StringUpper, Text, wwStringFromTime

System Functions

The following system functions in InTouch WindowMaker are supported by the ArcestrA Symbol Editor:

ActivateApp

Miscellaneous Functions

The following miscellaneous functions in InTouch WindowMaker are supported by the ArcestrA Symbol Editor:

DateTimeGMT, LogMessage, SendKeys, WWControl

Importing References

When you import a SmartSymbol, the following changes are made to tags and references:

| InTouch SmartSymbol | ArcestrA Symbol | Example |
|---------------------------|---|--|
| Local Tags | Prefixed with "InTouch:" keyword | Real Memory Tag "TankLevel1" is converted to "InTouch:TankLevel1" |
| Local Tags with dotfields | Prefixed with "InTouch:" keyword | Discrete Memory Tag "TankLevel1.InAlarm" is converted to "InTouch:TankLevel1.InAlarm" |
| SuperTags | Prefixed with "InTouch:" keyword. You need to manually enclose the expression by the following syntax: attribute("..."); | Real Supertag member "Reactor1\Level" is converted to "InTouch:Reactor1\Level". You need to change the expression manually as follows: attribute("InTouch:Reactor1\Level"); |
| I/O References | Prefixed with "InTouch:" keyword | Integer I/O Tag "Testprot:i00" is converted to "InTouch:Testprot:i00" |
| Galaxy References | "Galaxy:" prefix is removed | Galaxy Reference "galaxy:Pump1.Valve1" is converted to "Pump1.Valve1" |

The following items are imported with functional change:

| InTouch SmartSymbol | ArchestrA Symbol | Example |
|--|---|---|
| Galaxy:ObjectTagname. Property.#VString | "Galaxy:" prefix is removed but #VString is not supported. Applies also for #VString1, #Vstring2, #VString3 and #VString4 | "Galaxy:Tank.PV.#VString4" is converted to "Tank.PV" |
| Galaxy:ObjectTagname. Property.#ReadSts | "Galaxy:" prefix is removed but #ReadSts is not supported | "Galaxy:Tank.PV.#ReadSts" is converted to "Tank.PV" |
| Galaxy:ObjectTagname. Property.#WriteSts | "Galaxy:" prefix is removed but #WriteSts is not supported | "Galaxy:Tank.PV.#WriteSts" is converted to "Tank.PV" |
| Galaxy:ObjectTagname. Property.#EnumOrdinal | "Galaxy:" prefix is removed but #EnumOrdinal is not supported | "Galaxy:Selection.Sel1.#EnumOrdinal" is converted to "Selection.Sel1" |

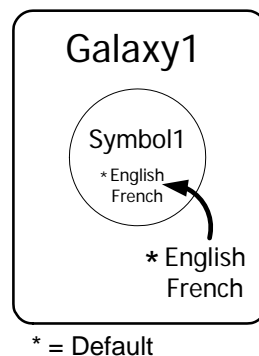
Chapter 15

Switching Languages for Graphic Elements

This section describes how to switch the language shown for ArcestrA symbols, the effects of different language settings between the Galaxy and an individual ArcestrA symbol, and language switching behaviors for certain features, such as embedded symbols, custom properties, and string substitution.

About Language Switching for ArcestrA Graphics

The language settings of the Galaxy control which languages are available to symbols. You cannot add a language at the symbol level. Languages are only added at the Galaxy level using the ArcestrA IDE.



Using the Symbol Editor, you select which of the Galaxy-configured languages you want to show for your symbol. By default, symbol text is shown in the default language and font for the Galaxy.

You export symbol text for translation using the IDE.

Graphic Elements that Support Translation

The following table describes the graphic elements that support translation. Element fonts are fonts set using the Property Editor and Symbol Editor tools.

| Graphic Element | Type of Translation Supported |
|------------------------|---|
| Text | Element Text, Font |
| TextBox | Element Text, TextFormat, Font |
| Button | Element Text, TextFormat, Font |
| Radio Button Group | Element Font |
| CheckBox | Element Font |
| EditBox | Element Font |
| ComboBox | Element Font |
| ListBox | Element Font |
| Calendar | Element Font |
| DateTime Picker | Element Font |
| Embedded Symbol | String substitutions and overridden custom properties defined as static strings |
| Client Control | Element Font |
| Base Symbol | Custom Properties defined as static strings |

Animations that Support Translation

The following table lists the user translation support for animations. When you switch the language at design time or run time, the following portions of the animation change accordingly.

| Animation | Type of Translation Supported |
|--------------------------------|---|
| ValueDisplay (Boolean Type) | Static string in the True Message of the Boolean type. Static string in the False Message of the Boolean type. |
| UserInput | Static string in the Message to User for all types. Static string in the True Prompt for Boolean type. Static string in the False Prompt for Boolean type. Static string in the True Message for Boolean type. Static string in the False Message for Boolean type. |
| ToolTip | Only the static string in the tooltip animation can be translated. |
| ShowSymbol | Only the static string in the title of the ShowSymbol animation can be translated when the Use Symbol Name for Window Title option is unchecked. |
| PushButton (String Type) | Only the static strings in the Value1 and Value2 of the String type Pushbutton animation can be translated. No other types of PushButton animations can be translated. |
| Radio Button | Only the captions of individual radio button items in the Static type, Array type, or Enum type RadioButton animations can be translated when the Use Values as Captions option is unchecked. |
| CheckBox | Only the caption string of the CheckBox animation can be translated when the Override the caption with the following expression option is unchecked. |

| Animation | Type of Translation Supported |
|-----------|---|
| EditBox | The configuration text of the EditBox animation can be translated only when the EditBox data source is left empty. If a data source is provided, the EditBox text cannot be exported for translation. |
| ComboBox | Only the captions of individual combo box items in the Static type, Array type, or Enum type ComboBox animation can be translated when the Use Values as Captions option is unchecked. |
| ListBox | Only the captions of the individual list box items in the Static type, Array type, or Enum type ListBox animation can be translated when the Use Values as Captions option is unchecked. |

Selecting the Language for a Symbol

When you select a language for a symbol, all graphic elements show the translated text associated with the selected language, if it is available. You can switch languages even if you open the symbol in read-only mode.

You can only select languages that are currently configured for the Galaxy.

Note: The current element creation font does not change when you switch to a different language.

To select the language for a symbol

- 1 Open the symbol in the Symbol Editor.
- 2 In the **Languages** panel, select the language from the list.

Removing a Language for a Symbol

If you remove a language from the Galaxy, the language is still available for a symbol until you specifically remove it.

You cannot remove a language from a symbol if the language is configured at the Galaxy level.

To remove the language for a symbol

- 1 Open the symbol in the Symbol Editor.
- 2 On the **Special** menu, point to **Configure** and then click **Locales**. The **Configure Languages** dialog box appears.
- 3 Select the language to remove and click **Remove**.
- 4 Click **OK**.

Creating Elements When Multiple Languages are Defined for a Galaxy

You must select the default language for the Galaxy before you create a new graphic element. You cannot create an element if you have a secondary language selected in the Symbol Editor. This includes:

- Duplicating an element
- Pasting an element
- Embedding a symbol
- Grouping or ungrouping elements
- Combining paths
- Breaking paths
- Import an InTouch SmartSymbol
- Convert a symbol to a group

Moving Symbols to Galaxies with Different Language Settings

If you import a symbol into a different Galaxy, any translation that already exists for the symbol is retained; however, only languages that are configured for the new Galaxy are available for use.

If you open a symbol that does not contain the default language of the Galaxy, then the default language is added to the list of languages for that symbol and is set as the new default language of the symbol. The text strings from the last saved default language of that symbol are transferred over to the new default language of the symbol.

For example, if a symbol only has the French language configured and is opened in a Galaxy where English is the default language, the French strings are transferred over to the English language when the symbol is opened, and English is made the default language for the symbol. The French language still exists for the symbol, but it will no longer be the default language.

If a previously imported symbol has English and French configured with French as the default language and the symbol is opened in a Galaxy with English as the default language, the default language for the symbol is set to English.

If you open a symbol in read-only mode, no changes are made to the symbol language settings to reflect the language settings of the Galaxy.

We recommend that you validate templates and instances that have symbols after migrating them to a new Galaxy and before exporting localizations.

How Fonts are Applied at Design Time

When you create a graphic element that supports visible text, it is created in the default language of the Galaxy. The font used is the last font persisted in the editor. However, if you provide a specific translation for an element in a secondary language, the Galaxy-configured font for the language is applied to the element.

For example:

- 1 You configure three languages: English (Default, Font = Arial), French (Font = Courier New) and German (Font = Times New Roman).
- 2 You open a symbol S1 in the English language. The editor default is Arial. You create a textbox in English. It is created with the Arial font.

- 3 You switch to German and translate the text. The font changes to Times New Roman, which is the font configured for German. The original font size and style remains the same.
- 4 You switch to French. The font is Arial because the text in French is not translated yet.

Language Switching for Embedded Symbols

When you embed a symbol into another symbol, any translations are also loaded for the embedded symbol. Switching the language for a symbol also switches the language for all embedded symbols.

For example, a symbol S1 contains a text graphic with the text "English String." English is the default language. The text is set to "French String" for the French language. The following steps describe language switching for the embedded symbol:

- 1 You embed the symbol S1 into the symbol S2.

You see the text showing the "English String."

- 2 You switch the language to French in the editor of S2.

The text in the embedded symbol S1 switches to "French String."

If you convert an embedded symbol to a group, any translations defined for the embedded symbol are migrated to the new elements created in the symbol. If the embedded symbol supports languages that are not defined in the base symbol, those translations are removed during the conversion.

If you open a symbol containing one or more embedded symbols and the current language of the Symbol Editor is not available in the embedded symbols, then the embedded symbols use the Galaxy-configured default language, if available. If the Galaxy-configured default language is not available, then the embedded symbols use the last saved default language of that symbol.

For example:

- 1 You create symbol S1 with French and German languages. The default language for the Galaxy is German.
- 2 You embed S1 inside S2 and then save and close S2.
- 3 You change the Galaxy default language to French.
- 4 You add English as another language.
- 5 If you open the symbol S2 in the English language, then the symbol S1 is shown using the French text. This is because French is the Galaxy-configured default language.

Embedded symbols support translations for custom properties. For more information on translating custom properties, see "Translating String Custom Properties" on page 433.

Embedded symbols also support translations for string substitutions. For more information on translation and string substitutions, see "String Substitutions and Language Switching" on page 432.

String Substitutions and Language Switching

You can substitute strings for textual elements within the symbol. For general information about string substitution, see "Substituting Strings" on page 174.

If you perform a first-time string substitution on an embedded symbol in the primary language, that substitution is shown in the secondary languages. You can then perform a substitution in the secondary language to create a string substitution specific to the secondary language.

If you perform a first-time string substitution on an embedded symbol in a secondary language, the substitution is also applied to the primary language, because the translated string that previously existed for the primary language is no longer valid. Because the primary language value is changed in the symbol, this string applies to all secondary languages configured. You can then perform a second substitution in the primary language, which will apply to all secondary languages except the ones that have had a specific substitution set.

If you perform a string substitution in a secondary language with an existing string substitution in the primary language, the new substitution is applied to the secondary language only.

The following design time and run-time rules are applied during a language switch to properly update an embedded symbol with the current substitutions for the language:

- 1 Apply the string substitutions from the default language.
- 2 Apply the string substitutions from the secondary language, if switching to a secondary language.

For example, an embedded symbol contains a text graphic with the text "English String." English is the default language. The following steps describe how changing the language affects string substitution for the embedded symbol:

- 1 While editing in the default language, you select an embedded symbol and open the string substitution dialog box.
You can see the old column with a value of "English String."
- 2 You replace the "English String" with "New English String."
- 3 You close the string substitution dialog box.

- 4 You switch to the French language.
- 5 You open the string substitution dialog box and see the string "New English String" in the old column.
- 6 You now replace the "New English String" with "New French String."
- 7 You close the string substitution dialog box.
- 8 You switch to the German language.
- 9 You open the string substitution dialog box and see the string "New English String" in the old column.
- 10 You now replace the "New English String" with "New German String."
- 11 You close the string substitution dialog box.
- 12 You switch to the French language.
- 13 You open the string substitution dialog box and see the string "New French String" in the old column.

If you select an alternate symbol, the string substitutions made on the initial symbol are reapplied to the new symbol across all languages.

The behavior of ArchestrA client control string substitutions are the same as the embedded symbols.

Translating String Custom Properties

You can translate custom properties that are defined as static strings.

If the custom property is configured with a reference, then that reference applies across all languages in the symbol. If you change that reference in any language to a static string, that string is set for all languages, and you can provide specific translations in the other languages.

For example, you create a custom property CP1 of type string with a default value of "Hello." You can now translate this custom property. You switch to another language in the editor and modify the default value of CP1 to UD1.str1 (changed from string to reference). Now CP1 cannot be translated. If you go back and change CP1 from a reference to a string, you can translate it again. The value you place in the default value is the value shown for all other languages if you do not specify a different string in that language.

When the custom property dialog box opens, it shows the appropriate translated values for the constant string custom properties, as determined by the translation precedence rules. For more information on these rules, see "Precedence Rules for Showing the Language and Font" on page 442.

Translating Custom Properties for a Base Symbol

New custom properties you create can be translated specifically for the symbol, provided that:

- They are configured as string custom properties.
- The default language value of the custom property is a static string (as opposed to a reference).

For example:

- 1 While editing the default language, you create a new custom property called CP1 with a data type of string.
- 2 You enter the value "English String" as the custom property value.
- 3 You close the custom property dialog box to save the changes.
- 4 You switch to the French language.
- 5 You open the custom property dialog box and see the string "English String."
- 6 You can now enter a specific string for French in the CP1 custom property. You cannot enter a reference in the French language.

Translating Custom Properties for an Embedded Symbol

You can translate custom properties exposed in an embedded symbol if the overridden value is a constant string. This also applies if the base value is a reference and is overridden to a constant string. All the translations for a single overridden custom property must be constant string values. The first localized overridden value in one local is propagated to all the other locales as the default overridden value.

For example:

- 1 While editing the default language, you select an embedded symbol and open the custom property dialog box.

You can see a constant string custom property called "CP1" with a value of "English String."
- 2 You close the custom property dialog box.
- 3 You switch to the French language.
- 4 You open the custom property dialog box and see the string "English String."
- 5 You can now enter, for example, "Overridden French String" string for French in the CP1 custom property. You cannot enter a reference in the French language.
- 6 If you switch to English Language, you see "Overridden French String" as the default overridden value.

Translation Support for Client Controls with Satellite Assemblies

A satellite assembly is a .NET Framework assembly that contains resources specific to a given language. You can place the resources for different languages in different assemblies, and the correct assembly is loaded into memory if the run-time user views the application in that language.

If you import a client control with satellite assemblies, then the satellite assemblies are also imported.

The translation support for client controls with satellite assemblies is as follows:

| Scenario | Behavior of Client Control |
|-----------------------------|--|
| Design Time - Symbol Editor | The ArcestrA Symbol Editor only supports English. The application locale has no effect on how the client controls are shown. |
| Design Time - WindowMaker | The localized version of the InTouch HMI being used determines how the client controls are shown. |
| Run Time - WindowViewer | The localized version of the InTouch HMI being used determines how the client controls are shown. The application locale has no effect on how the client controls are shown. |

When you export text for translation, the base font information for client controls is not included. You need to use satellite assemblies to control this information.

Translation Support for ArcestrA Client Controls

ArcestrA client controls, such as the ArcestrA Alarm Control, support the following types of translation:

- ArcestrA client controls with satellite assemblies work similar to other types of client controls.
- ArcestrA client controls support translated string substitutions. For more information on translated string substitution, see "Switching Languages and String Substitutions at Run Time" on page 444.

The number and type of strings used by the ArcestrA client control is specific to the control.

When you export text for translation, the base font information for ArcestrA client controls is not included. You need to use satellite assemblies to control this information.

Importing InTouch SmartSymbols that Have Translated Data

You can import InTouch SmartSymbols having translated data into the Symbol Editor. All the language data in a SmartSymbol is imported into the ArcestrA symbol, including:

- Configured languages.
- Translated content for graphic elements (text and font)
- Translated animation links.

After you import, you can view the language data in the ArcestrA Symbol Editor at design time or in InTouch WindowViewer at run time.

For example, you import a SmartSymbol with language data for French and German. After the import, the French and German languages are added to the ArcestrA Symbol, if they do not already exist.

The language switching behavior at run time matches the behavior of the original SmartSymbol. Any variation in the default font usage at run time between ArcestrA Symbols and native InTouch graphics is resolved during the SmartSymbol import. During the import, elements that have translations but not a translated font are detected and their font set to the Galaxy-configured font for that language.

Support for Empty Strings

You cannot substitute an empty string in the primary locale. You must use space characters. When you set an empty string for a primary locale, the empty string is propagated to all other locales that do not have translations.

Performing a first time substitution of an empty string in a secondary locale puts a space character in the primary and the current locale. The remaining locales will match the primary value if they do not already have a specific value.

If a primary locale contains an empty string, it will be exported for translation.

If you substitute an empty string for a secondary locale, the element shows as empty. However, if you switch to the primary locale and then back to the secondary locale, the element shows the primary string substitution again.

Language Switching Example

The following table describes the effects of language switching on the various parts of the system. In this example:

- The Galaxy is configured with two languages: English and French.
- The default language in the Galaxy is English.
- There are various symbols configured in the Galaxy, some of which contain partial or mismatched language configurations compared to the Galaxy's configured languages.

| Action | Effect on the Languages Configured for the Symbol | Effect on Elements that contain English Translations | Effect on Elements that contain French Translations |
|--|--|--|--|
| You open a new symbol. | English language is added to the symbol. | None | None |
| You open an existing symbol that only has English defined. | None | None | None |
| You open an existing symbol that only has the French language defined. | English language is added to the symbol. | French strings are transferred into the English language. | French strings are marked as specific translations for French. |
| You open an existing symbol that only has the German language defined. | English language is added to the symbol. | German strings are transferred into the English language. German strings are marked as specific translations for German. | None |
| You change a text string in the English language. | None | New string is set for the English language. | None |

| Action | Effect on the Languages Configured for the Symbol | Effect on Elements that contain English Translations | Effect on Elements that contain French Translations |
|--|--|--|---|
| You switch to the French language for the first time in a symbol that only had the English language. | French language is added to the symbol. | None | The default language strings are shown unless a specific French translation exists. |
| You change a string while viewing the French language. | None | None | The new string is set as the specific translation for the French language and used for display. |
| You create a new text element in a symbol that has English only while viewing the English language. | None | The new element's string value is saved for translation. | None |
| You delete an element with translations in English and French. | None | English translations are removed. | French translations are removed. |
| You copy animations from an element. | None | English animation translation strings are put into the clipboard. | French animation translation strings are put into the clipboard. |
| You paste animations to an element. | None | English animation translation strings are put into the destination animations. | French animation translation strings are put into the destination animations. |
| You clear animations for an element. | None | English animation strings are removed. | French animation strings are removed. |
| You copy and paste elements from a German-only symbol into a symbol containing English and French. | None | German strings are placed in the English language. | None |

| Action | Effect on the Languages Configured for the Symbol | Effect on Elements that contain English Translations | Effect on Elements that contain French Translations |
|--|--|---|--|
| You copy and paste elements from an English and German symbol into a symbol containing English and French. | None | English strings from the source symbol are copied during the paste. The German strings are dropped. | None |
| You export the French language, having never switched to the French language. | The French language is added to the symbol for the purposes of the export. The language is not saved back to the symbol during the export. | English strings are exported in the "Phrase" XML attribute field. | If specific strings exist for French, they are exported in the "Translation" XML field. |
| You import the French language. | The French language is added. | None | Any translations provided in the import are marked as specific translations for the French language. If the translation is empty, the default language value is shown. |
| You convert a symbol to a group. | Same logic as copy/paste. | Same logic as copy/paste. | Same logic as copy/paste. |
| You delete the German language. | German is removed. | None | None |

Overriding Translated Strings for ArcestrA Symbols in WindowMaker

After you embed an ArcestrA symbol into an InTouch window, you can override the translations for:

- Strings on the substitutable graphic elements within the ArcestrA symbol.
- String type custom properties on the ArcestrA symbol.

To translate these overrides, you must export and import the strings for the managed InTouch application.

Overriding Translated String Substitutions

After you export the managed InTouch application using the IDE, all the string substitution overrides are exported into an InTouchViewApp dictionary file.

After you import the translated InTouchViewApp dictionary file, you can view the translated string overrides in InTouch WindowViewer at run time.

Overriding Translated Custom Properties

After you export the managed InTouch application using the IDE, all the custom property overrides are exported into an InTouch ViewApp dictionary file.

After you import the translated InTouchViewApp dictionary file, you can view the translated custom properties in InTouch WindowViewer at run time.

Language Switching at Run Time

At run time, languages can be switched in the following ways:

- Using the InTouch script function `SwitchDisplayLanguage()` from an InTouch script only.
- Setting the value of the `$Language` system tag within an InTouch or ArcestrA script.
- Selecting the language from the **Special** menu. For managed InTouch applications, the list of languages shown is based on the languages configured in the Galaxy.

How Languages are Shown in WindowViewer

At run time, ArcestrA symbols are shown in the language set for InTouch WindowViewer.

If the ArcestrA symbol's language does not match the WindowViewer language setting, the language shown is the language that was the default language for the symbol when it was last saved.

For example, an InTouch Window has three different embedded ArcestrA symbols: S1, S2, and S3.

- S1 has a Text graphic with French and German languages configured.
- S2 has Textbox graphic with only German language configured.
- S3 has a Button graphic with English and Spanish languages configured. S3 has two client scripts. The first script switches the language to Chinese, and the second script switches the language to Spanish.

English is the default language. The following steps show how the translated symbols are shown:

- 1** Set the language to French in the WindowViewer.

The Text in S1 shows the "French String."

The Textbox in S2 shows the "German String."

The Button in S3 shows the "English String."

- 2** Run the first client script on symbol S3.

The Button in S3 still shows "English String."

- 3** Run the second client script on symbol S3.

The Button in S3 shows "Spanish String."

- 4** Set the language to German in the WindowViewer.

The Text in S1 shows the "German String."

The Textbox in S2 still shows the "German String."

The Button in S3 shows the "English String."

Precedence Rules for Showing the Language and Font

All elements and animations that can be translated follow these precedence rules for showing the translated text:

- 1** The default language holds the master language text for the translation.
- 2** The secondary languages use the default language's text if no specific translation exists for the secondary language.
- 3** If a specific translation value exists for a secondary language, that text is shown instead of the default language's text.

The precedence rules for which font to use for a graphic element that supports visual text are as follows:

- 1** If there is no text translation for an element, the element shows the text in the default language.
- 2** If there is a specific translation in the secondary language and no translated font specified for the element, the element shows text using the font specified in the Galaxy for that language.
- 3** If there is a specific translation in the secondary language and a specific font set for the element, the element shows the text using the specific font.

For example:

- 1** You create a TextBox element on a symbol with the text "English String." The Galaxy-configured languages are English with Arial font and German with Tahoma font.
- 2** You switch over to the German Language. The TextBox element still shows "English String" with the Arial font.
- 3** You change "English String" to "German String." Because there is now a translation for this string, the TextBox shows "German String" with the Tahoma font.

Default Language Fonts at Run Time

InTouch WindowViewer uses the default font for a given language if an element has a translation but no translated font specified. For ArcestrA symbols, the font element of the default Galaxy language is used.

Switching Languages for Custom Properties at Run Time

Embedded ArcestrA symbols and ArcestrA symbols embedded in the InTouch HMI can contain translated custom properties.

When you switch to a language for the first time, the translated overrides of the custom properties are loaded from the dictionary file of the ArcestrA symbol's overrides.

After switching to a new language in the InTouch WindowViewer, the translated custom properties are updated based on the switched language. All the animations/scripts that subscribe to a custom property are notified of the value change and the animation/script values are updated accordingly.

For example:

- Symbol S1 has a constant string custom property CP1.
- S1 has two languages configured: English and French. English is the default language. The English is "English CP1."
- CP1 has the French translation, "French CP1." S1 has two graphic elements: Textbox and Button.
- Each graphic has a value display animation that subscribes to CP1.
- An InTouch Window has two embedded symbols of S1.
- The second instance of S1 has the translated CP1 override in French ("French Override CP1") and German ("German Override CP1").

The following steps describe how the translated custom properties are shown.

- 1** Set the language to French in WindowViewer.

The value display animation of Textbox and Button in the first instance of S1 shows "French CP1."

The value display animation of Textbox and Button in the second instance of S1 shows "French Override CP1."

- 2** Set the language to German in the WindowViewer.

The value display animation of Textbox and Button in the first instance of S1 shows "English CP1."

The value display animation of Textbox and Button in the second instance of S1 shows "German Override CP1."

Switching Languages and String Substitutions at Run Time

An embedded ArcestrA symbol and ArcestrA symbol embedded into InTouch can contain translated string substitutions. All the graphic elements that contain the translated string substitutions are updated based on the current language setting.

For example:

- Symbol S1 has two graphic elements: Textbox and Button. S1 has two languages configured: English and French. The strings are "English String" and "French String." English is the default language.
- The Textbox element has a French string substitution, "French Sub String." The Button element has an English substitution, "English Sub String."
- An InTouch window has two embedded symbols of S1. The second instance of S1 has the translated string substitution overrides for Textbox and Button in French ("French Override Sub String") and German ("German Override Sub String").

The following steps describe how the translated string substitutions are shown.

- 1** Set the language to French in WindowViewer.
 - The text of Textbox and Button in the first instance of S1 shows "French Sub String" and "English Sub String," respectively.
 - The Text of Textbox and Button in the second instance of S1 shows "French Override String."
- 2** Set the language to German in the WindowViewer.
 - The Text of Textbox and Button in the first instance of S1 shows "English String" and "English Sub String," respectively.
 - The Text of Textbox and Button in the second instance of S1 shows "German Override String."

Language Settings for Popup Symbols

At run time, all popup symbols are shown using the current language setting for WindowViewer. The popup symbol always tries to show the current language, even if the symbol opening the pop-up does not support the language.

For example, a symbol S1 is configured with French and German languages. The following steps show how the current language setting updates:

- 1 Switch the language from French to German in WindowViewer.

The current language setting is to German.

- 2 Open a popup symbol S2 from the symbol S1.

The symbol S2 is shown using the German language. If symbol S2 does not have German language configured, then it is shown using its last saved default language.

- 3 Close the popup symbol S2.

- 4 Changes the Language property of the symbol S1 to French. The current language setting of the InTouch WindowViewer is now German, and the language setting of the symbol S1 is French.

- 5 Open a popup symbol S2 from the symbol S1.

The symbol S2 is shown using the French language. If symbol S2 does not have French language configured then it is shown using its last saved default language.

Dynamic Propagation of Language Changes to Popup Symbols

When you change the language setting of WindowViewer, the change dynamically propagates to all popup symbols. This includes both modal and modeless popup symbols.

For example, symbol S1 is configured with French and German languages. WindowViewer is set to show the French language. The following steps show how the language is updated for the popup symbols:

- 1 You open a modeless popup symbol S2 from the symbol S1.
- 2 You open a modeless popup symbol S3 from the symbol S2.
- 3 You open a modeless popup symbol S4 from the symbol S1.

The symbols S1, S2, S3, and S4 are shown using the French language.

- 4 You change the language to German.

The symbols S1, S2, S3 and S4 now show the German language. If the popup symbol does not support the German language, it is shown using the last saved default language of that popup symbol.

Language Settings and Data Types

All data and symbols are shown by ArchestrA as English formatted data types, regardless of the language being shown, the operating system on which the InTouch HMI is installed, or the translated version of the InTouch HMI you are using.

Chapter 16

Working with the Show/Hide Graphics Script Functions

The Show/Hide Graphic script functions enable you to write ArcestrA Graphics scripts to display a symbol as a pop-up window and close the pop-up window.

In this section, you will find the following topics:

- About the Show/Hide Graphic Functions
- Configuring the Show/Hide Graphic Script Functions
- Using the Show/Hide Script Parameters and Properties
- Run Time Behavior of the Show/Hide Graphic Functions
- Show/Hide Graphic Script Tips and Examples

About the Show/Hide Graphic Functions

The Show/Hide Graphics script functions are in addition to the Show/Hide Symbol animation feature, which enables you to display a symbol as a pop-up window through symbol 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 457.

Like the Show/Hide Symbol animation feature, you can control the properties of the symbol through the Show Graphic feature. You can configure the script to specify:

- Which symbol 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 symbol that you want to display.
- Values assigned to a symbol's custom properties.

You can use the HideSelf script function for ArcestrA 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 InTouch applications only.

Configuring the Show/Hide Graphic Script Functions

You must first include a script that contains the ShowGraphic function to display a symbol 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 ArcestrA symbol, displayed through the ShowGraphic script function. The HideSelf script function allows you to close the symbol, 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 ArcestrA Graphic being used in the InTouch application.

To include a script that contains the Show/Hide Graphic functions within a symbol animation action script

- 1 Open the ArcestrA IDE.
- 2 Create a symbol or open an existing symbol.

- 3 Draw a graphic, and then double-click it to open the **Edit Animations** page.
- 4 Open the action script editor.
- 5 Click the **Display Script Function Browser** icon. The **Script Function Browser** appears.
- 6 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 );
```

Note: You can click **Help** to view the Help after you have selected any Graphic Client script function.

- 7 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 449.

For details on the scripts and samples, see "Show/Hide Graphic Script Functions Guidelines" on page 450.

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. The **Galaxy Browser** page 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. The **Galaxy 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
- Show/Hide Graphic Script Tips and Examples

For information about script syntax and parameters along with basic script examples, see "HideGraphic()" in Chapter 2, "QuickScript .NET Functions," in 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
- Height and Width Aspect Ratio
- Incompatible GraphicInfo Properties

Using the Identity Property in the ShowGraphic() Function

The Identity must be unique across the InTouch 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: Symbol 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: Symbol 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 symbol 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>. <Symbol 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 | |

| Window Relative Position | Size: Relative To | Incompatible Properties | Notes |
|---------------------------------|---------------------------|--------------------------------|---------------------------------|
| Desktop | CustomizedWidth Height | X Y ScalePercentage | |
| Window | Graphic | X Y Width Height | |
| Window | Desktop | X Y Width Height | RelativeTo should be Window |
| Window | CustomizedWidth Height | X Y ScalePercentage | |
| ClientArea | Graphic | X Y Width Height | |
| ClientArea | Desktop | X Y Width Height | RelativeTo should be ClientArea |
| ClientArea | CustomizedWidth Height | X Y ScalePercentage | |
| ParentGraphic | Graphic | X Y Width Height | |

| Window Relative Position | Size: Relative To | Incompatible Properties | Notes |
|---|------------------------------|--|--|
| ParentGraphic | Desktop | X Y Width Height | RelativeTo should be ParentGraphic |
| ParentGraphic | CustomizedWidth Height | X Y ScalePercentage | |
| ParentElement | Graphic | X Y Width Height | |
| ParentElement | Desktop | X Y Width Height | |
| ParentElement | CustomizedWidth Height | X Y ScalePercentage | |
| Mouse | Graphic | X Y Width Height StretchWindow ToScreenWidth StretchWindow ToScreenHeight | |

| Window Relative Position | Size: Relative To | Incompatible Properties | Notes |
|---------------------------------|---------------------------|--|--|
| Mouse | Desktop | X Y Width Height StretchWindow ToScreenWidth StretchWindow ToScreenHeight | RelativeTo should be Desktop |
| Mouse | CustomizedWidth Height | X Y Width Height StretchWindow ToScreenWidth StretchWindow ToScreenHeight | |
| DesktopXY | Graphic | Width Height StretchWindow ToScreenWidth StretchWindow ToScreenHeight | |
| DesktopXY | Desktop | Width Height StretchWindow ToScreenWidth StretchWindow ToScreenHeight | Conflicting WindowRelative Position and RelativeTo combination |
| DesktopXY | CustomizedWidth Height | ScalePercentage StretchWindow ToScreenWidth StretchWindow ToScreenHeight | |

| Window Relative Position | Size: Relative To | Incompatible Properties | Notes |
|---------------------------------|--------------------------|--|--|
| WindowXY | Graphic | Width Height StretchWindowToScreenWidth StretchWindowToScreenHeight | |
| WindowXY | Desktop | Width Height StretchWindowToScreenWidth StretchWindowToScreenHeight | Conflicting WindowRelative Position and RelativeTo combination |
| WindowXY | CustomizedWidth Height | ScalePercentage StretchWindowToScreenWidth StretchWindowToScreenHeight | |
| ClientAreaXY | Graphic | Width Height StretchWindowToScreenWidth StretchWindowToScreenHeight | |
| ClientAreaXY | Desktop | Width Height StretchWindowToScreenWidth StretchWindowToScreenHeight | Conflicting WindowRelative Position and RelativeTo combination |
| ClientAreaXY | CustomizedWidth Height | 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 an InTouch pop-up window.
- You can configure a symbol 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 and close the graphic from across symbols and across InTouch windows. You can manage the graphic across the entire InTouch 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 Symbol" on page 458.
- You cannot use the **Close Window** dialog box of InTouch WindowViewer to close the pop-up windows displayed by the ShowGraphic script function. For more information, see "Closing a Symbol" on page 458.
- Any graphic displayed by ShowGraphic script function or ShowSymbol animation always remains in front of InTouch windows, except InTouch pop-up windows. Even if you click an InTouch window, the window remains behind these graphics.
- Enabling in-memory graphics caching in WindowViewer 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 symbol with the ShowGraphic script function or with ShowSymbol animation, WindowViewer will perform a memory health check.

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 symbol is modal to the modal symbol behind it, calling the ShowGraphic function cannot change this symbol to be modeless to the current modal symbol. For more information, see "Working with Modal Windows" on page 461.

Closing a Symbol

You can close a symbol, 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 WindowViewer. You cannot close the graphic by closing the InTouch window or the symbol 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 the WindowViewer **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
- Working with Modal Windows
- Using Hierarchical References and Containment Relationships
- Scripting the Owning Object
- Assigning Custom Property Values of a Symbol

Using Predefined and Named Scripts

You can use the Show/Hide Graphic script functions inside container scripts. Container scripts refer 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 370, "Configuring the Predefined Scripts of a Symbol" on page 376, and "Adding Named Scripts to a Symbol" on page 378.

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 symbol, such as a pump, called "symbol01" and another "symbol02".
- 2 Add a button named "Close" to symbol01 on the graphic editor canvas, and add an action script to the button:

```
HideSelf();
```

- 3 Add a button named "Show Pump" in symbol02 on the 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 symbol, 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 symbol 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 symbol 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 symbol 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 symbol S3 closes and the S2 script completes.
- You cannot close S2 using a ShowGraphic (S2) script function with the same Identity until the modal symbol S3 closes and the S2 script completes.
- You can close S1 using a HideGraphic (S1) script function from another window because the subsequent symbol S2 is modeless.

Example 2: Modeless Symbol1 (S1) opens modal Symbol2 (S2) using a ShowGraphic() script function. Modal symbol 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 symbol S3 closes.

- You cannot close S1 using a HideGraphic (S1) script function from another window until the modal symbol S2 closes.
- You cannot close S2 using a ShowGraphic (S2) script function with the same Identity until the modal symbol S3 closes.
- You cannot close S1 using a ShowGraphic (S1) script function with the same Identity until the modal symbol 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 symbol by opening another symbol with the same Identity, effectively replacing the original symbol, we recommend that you do not use the same Identity as a symbol opened with ShowGraphic in a modal dialog.

Example 3: With the Graphic Cache memory feature enabled, a ShowSymbol (SS) has a button to show symbol Symbol1 (S1) (modal), and another button to show symbol Symbol1 (S1) (modeless). S1 is configured to open Symbol2 (S2) with the ShowGraphic() function. In this scenario:

- Click show symbol 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 symbol 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 458.

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 + ".SymbolName";</pre> |
| <pre>GraphicName = "MyArea.SymbolName";</pre> | <pre>GraphicName = MyArea.Tagname + ".SymbolName";</pre> |

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 symbol S1.

When writing scripts that invoke different owing objects at run time, create a separate InTouch window. Use this window as a container for all the symbols you will be associating with the different owing objects at run time.

This special InTouch window does not have to be used or become visible. This window merely functions as container to push all required symbols to the run-time nodes, and will ensure that the correct symbol is available to other InTouchViewApp objects when called by scripts.

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 symbol S1.

The symbol (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 (GraphicName="me.S1" or "Reactor_001.S1", and OwingObject = "Reactor_002") and execute the script at run time, the system displays Reactor_001.S1, though the relative reference within this symbol points to Reactor_002 object.

In such a case:

- The symbol 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 symbol name. If a relative reference is used in the GraphicName property, then the symbol will always open from only the host automation object instance, here "Reactor_001". If an absolute reference or Graphic Toolbox symbol name is used in the GraphicName property such as "Reactor_001.S1" or "S1", then the system will search for such symbol 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.

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 in WindowViewer, 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:

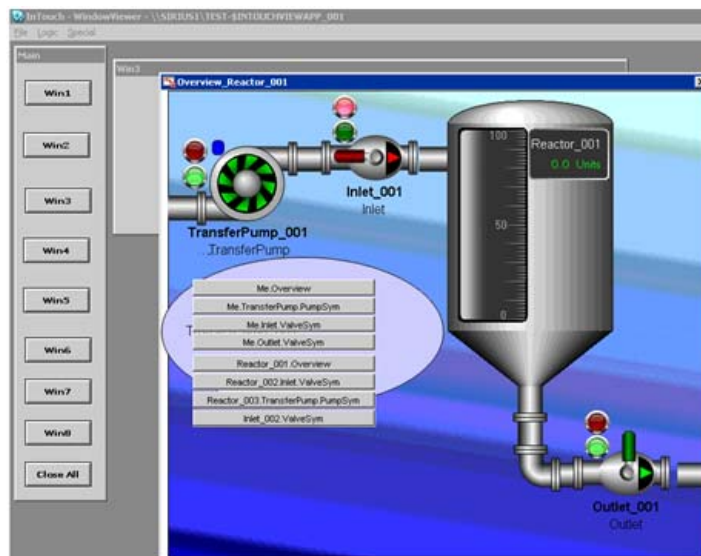
```
Dim graphicInfo as aaGraphic.GraphicInfo;  
graphicInfo.Identity = "Overview_" + Cp1;  
graphicInfo.GraphicName = "Reactor_001.Overview";  
graphicInfo.OwningObject = cp1;  
ShowGraphic( graphicInfo );
```

Where cp1 = "Reactor_001", "Reactor_002", or "Reactor_003". Object Reactor_001, Reactor_002, and Reactor_003 derive from the same template \$Reactor.

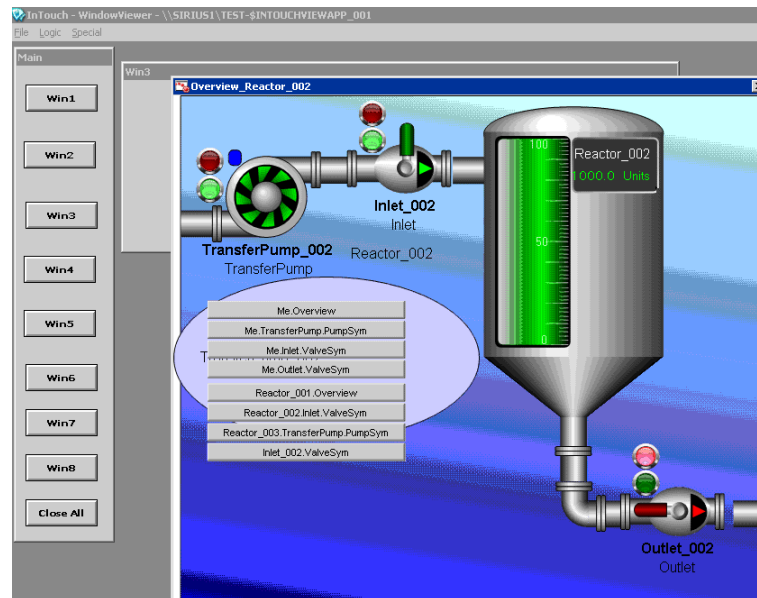
At run time, click the **ShowGraphicOverview** pushbutton to open the graphic. You can change **Reactor_001** to **Reactor_002** to switch between the two graphics, illustrated in the following sequence.



Displaying "Reactor_001":



Switched to display "Reactor_002" using the configured button:



Owning Object Scenario 2

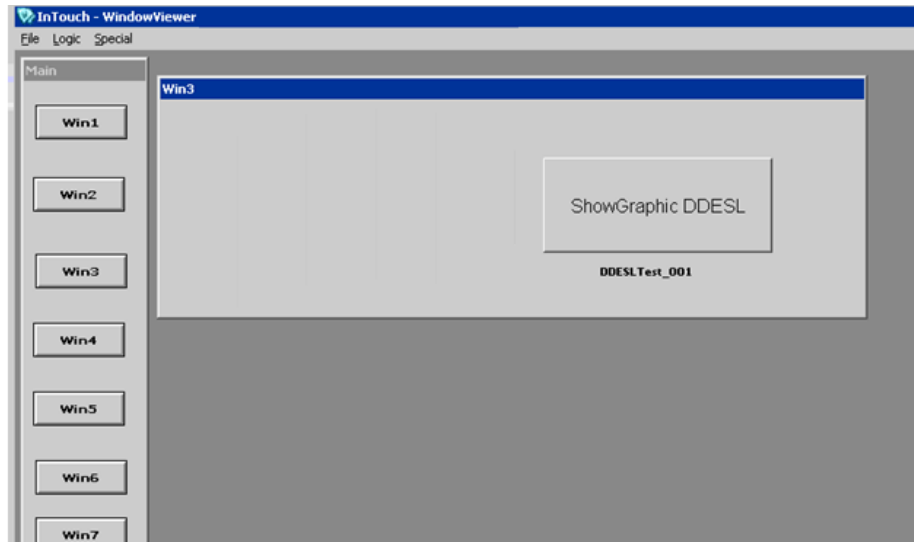
You need to monitor and maintain connection to several different data acquisition servers, but do not want to keep all server property windows open at all times. You can use `ShowGraphic()` with the `OwningObject` feature to switch back and forth between server status displays using a single interface element (pushbutton).

You can use the following script for the purpose:

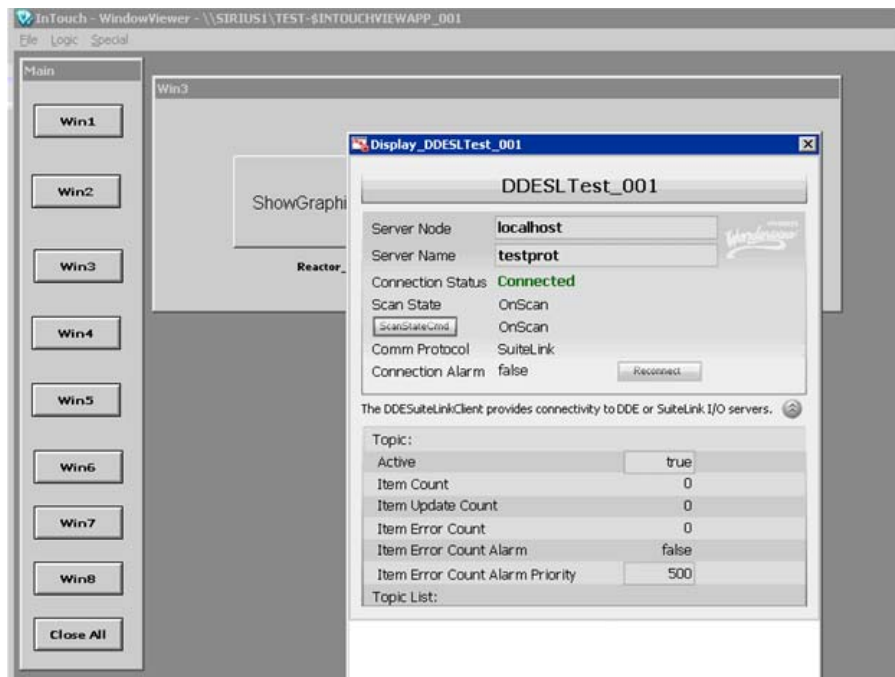
```
Dim graphicInfo as aaGraphic.GraphicInfo;
graphicInfo.Identity = "Display_" + cp2;
graphicInfo.GraphicName = "DDESL";
graphicInfo.OwningObject = cp2;
ShowGraphic( graphicInfo );
```

Where `cp2` = "DDESLTest_001", "DDESLTest_002", and DDESL is the Graphic Toolbox symbol. DDESLTest_001 DDESLTest_001 are the automation object instances.

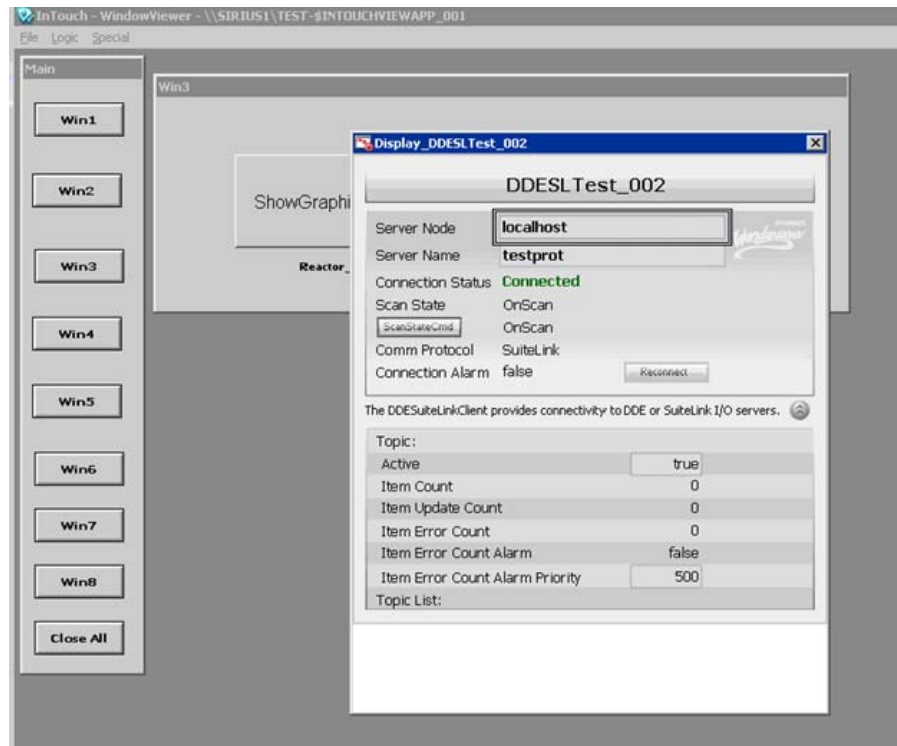
At run time, click the **ShowGraphicDDESL** pushbutton to open the graphic. You can change **DDESLTest_001** to **DDESLTest_002** to switch between the two graphics, illustrated in the following sequence.



Displaying server DDESL Test_001:



Switched to display DDESL Test_002 using the configured button:



Assigning Custom Property Values of a Symbol

Custom properties of a symbol can be set to values when a symbol 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 symbol "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 );
```

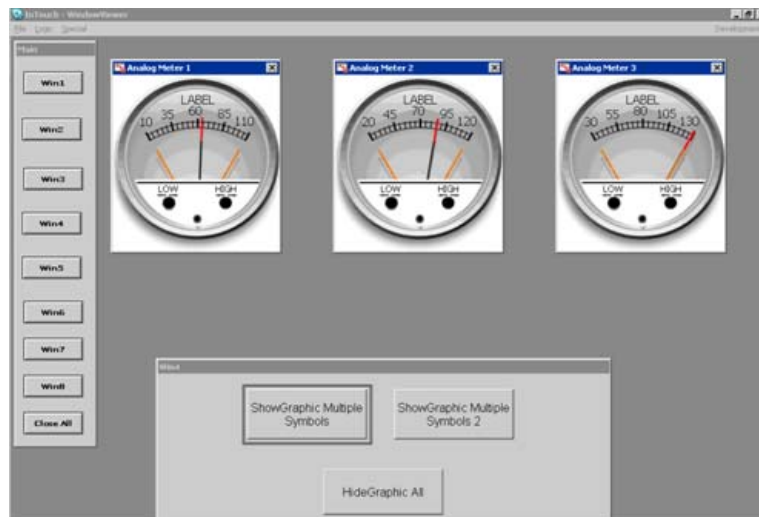
```
graphicInfo.Identity = "i2";
graphicInfo.GraphicName = "AnalogHiLo";
graphicInfo.HasTitleBar = true;
graphicInfo.WindowTitle = "Analog Meter 2";
graphicInfo.Resizable = true;
graphicInfo.WindowLocation = aaGraphic.WindowLocation.Center;
```

```
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 461.

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.

For details on index positions, see "HideGraphic()" in Chapter 2, "QuickScript .NET Functions," in the *Application Server Scripting Guide*.

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 );

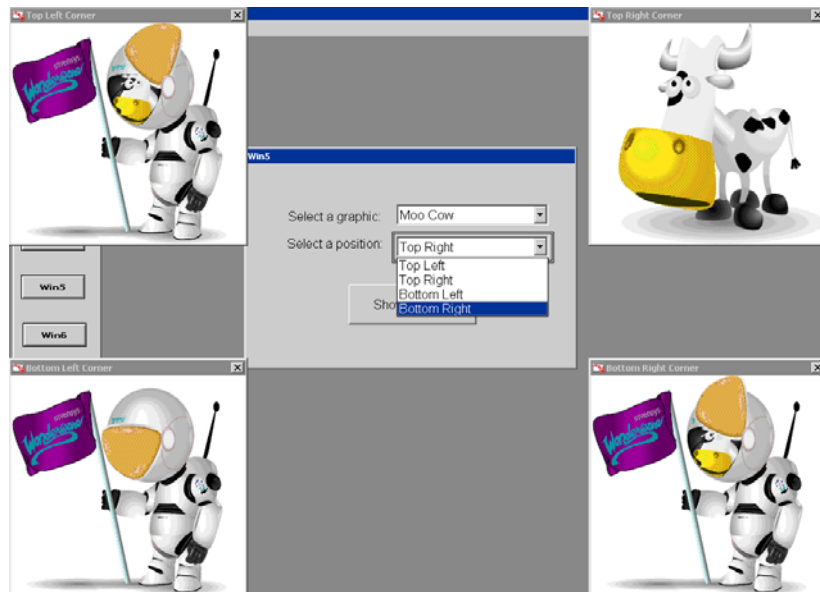
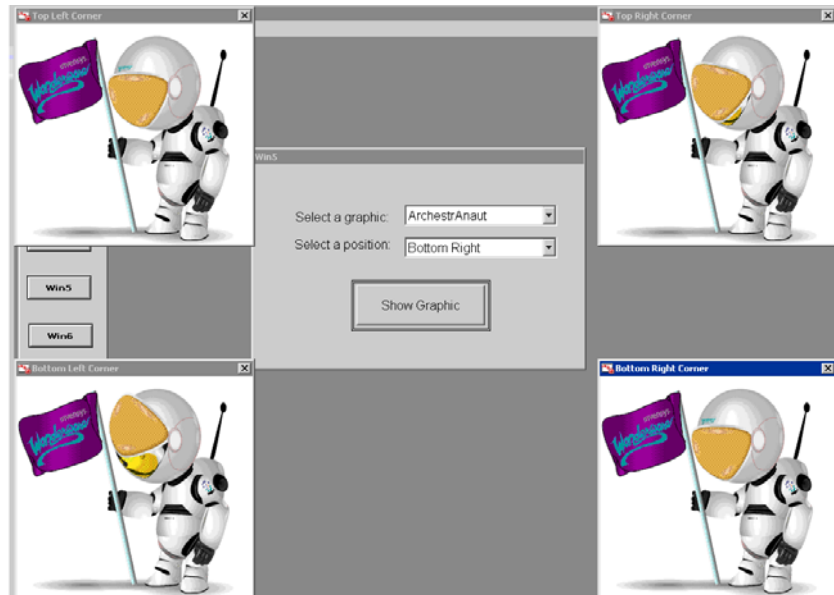
popup.WindowLocation = MyInt;

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 Symbols Scenario 3

You want the symbol 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 Symbols Scenario 4

You want the symbol in relative position with the window.

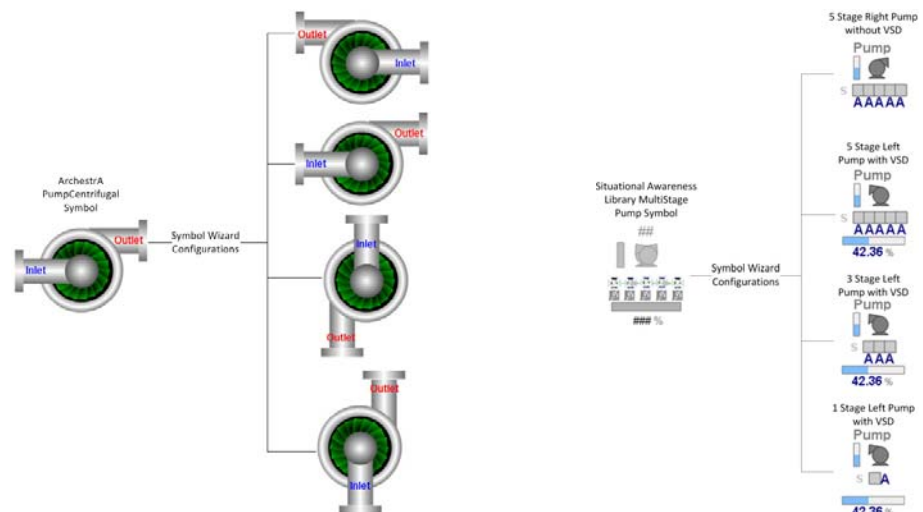
You can use the following script for the purpose:

```
Dim graphicInfo as aaGraphic.GraphicInfo;  
graphicInfo.Identity = "i1";  
graphicInfo.GraphicName = "S1";  
graphicInfo.RelativeTo= aaGraphic.RelativeTo.Window;  
ShowGraphic(graphicInfo);
```

Chapter 17

Working with Symbol Wizards

The ArchestrA Symbol Editor includes the Symbol Wizard Editor, which can be used to create reusable configurable symbols called Symbol Wizards. For example, a single ArchestrA 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 symbol. Situational Awareness Library symbols are designed using the Symbol Wizard Editor. However, they are protected symbols 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 symbol's design.

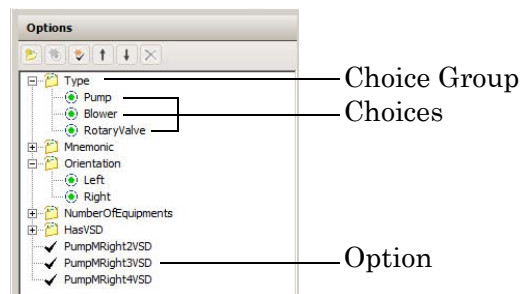


Incorporating multiple configurations in a single symbol reduces the number of symbols needed to develop an ArchestrA application.

Understanding the Symbol Wizard Editor

After enabling the Symbol Wizard Editor, the Symbol 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 symbol.
- The tabbed **Options** pane shows a hierarchical list of Choice Groups, Choices, and Options that define symbol 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 symbol layers. Beneath each layer, separate folders contain the symbol's graphic elements, custom properties, and named scripts associated with the layer. A symbol's graphic elements, custom properties, and named scripts are assigned to symbol 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 symbol 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 symbol.

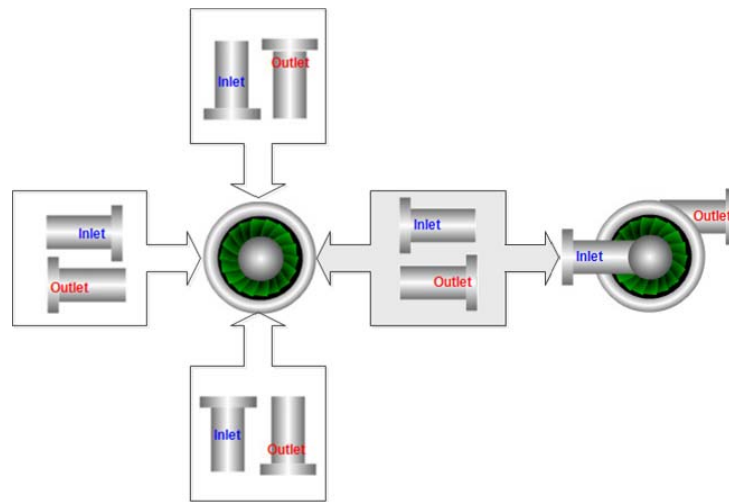
An item shown in the **Options** view list can be moved by selecting it, and then clicking the **Up** or **Down** arrow. If no Choice is specified as the default value for a Choice Group, the first Choice added to the Choice Group is always the default value.

In the example of an ArcestrA centrifugal pump symbol, 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 symbol 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 an ArcestrA centrifugal pump symbol, 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 symbol's configuration.



The blade housing does not belong to a layer because it is common to all pump symbol configurations. Graphic elements of a symbol that do not belong to a layer appear in all symbol 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 symbol 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 symbol configuration.

Associating graphic elements, named scripts, and custom properties to symbol layers involves working with the Symbol Wizard Editor **Layers** pane shown to the left of the graphic canvas.

Defining Symbol Configuration Rules

A rule defines an expression that determines if a given choice or option and its associated symbol 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"> • Period (.) A period concatenates a Choice Group to a Choice in a hierarchical expression. • Pipe () A pipe evaluates to a Boolean OR. • Ampersand (&) An ampersand evaluates to a Boolean AND. • Exclamation point (!) An exclamation point evaluates to a Boolean NOT. • Parentheses () A compound expression enclosed within parentheses is evaluated before other expressions in a rule. |

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 hierarchical 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 Symbol 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 494.

Designing a Symbol Wizard

The process of creating and implementing a Symbol Wizard has two workflows, a Designer workflow and a Consumer workflow:

- A Designer uses the Symbol Wizard Editor to create Symbol Wizards containing multiple configurations.
- A Consumer embeds a Symbol Wizard and then configures it for use in a managed InTouch application.

Creating Symbol Choice Groups, Choices, and Options

The following list summarizes the tasks that need to be completed by a Designer to create a Symbol Wizard containing multiple configurations.

- Define a symbol's Choice Groups, their Choices, and Options
- Assign rules to Choice Groups, Choices, and Options
- Associate graphic elements, custom properties, and named scripts to symbol layers
- Verify each symbol configuration with Symbol Wizard Preview

After planning the possible configurations for a symbol, Designers should know the properties and the possible attributes associated with each configuration. Designers create Choice Groups, Choices, and Options to define a symbol's properties and attributes.

Important: Situational Awareness Library symbols have predefined Choice Groups, Choices, and Options.

To create symbol choice groups, choices, and options

- 1 In the ArcestrA IDE, create a copy of a symbol in the Graphic Toolbox that 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 symbol in the Symbol Editor's canvas drawing area.
- 3 Click the Symbol Wizard icon shown on the Symbol Editor menu bar.

You can also show Symbol Wizard by pressing Alt+W or selecting it as an option from the **View** menu.

The Symbol 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 symbol 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 symbols 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 symbol 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 symbol's configurations.

Assigning Symbol Configuration Rules

Designers can specify rules for a symbol's defined Choices and Options. Choice Groups should not be included in symbol configuration rules.

These rules determine the graphic elements, custom properties, and scripts that belong to a symbol configuration. For more information about rule syntax, see "Defining Symbol Configuration Rules" on page 478.

To define symbol configuration rules

- 1 Show the selected symbol in the Symbol 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 symbol configurations. Choices without rules are always visible.

4 Repeat steps 2-3 to specify rules for the remaining Choices of the symbol.

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 symbol's default configuration or not.

8 In the **Description** field, enter a description of the Option.

The description appears when the Consumer embeds the symbol 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 symbol.

Updating Symbol Layers

Symbol Wizard automatically creates a set of default layers that match the hierarchical set of Choices and Options defined for a symbol. Each Choice layer has an assigned default rule containing the expression *ChoiceGroup.Choice* that defines an attribute of a symbol'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.

Designers 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 symbol layers may not be necessary if the default set of layers created for Choices and Options can create all symbol configurations.

If a symbol 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 symbol layer

1 Show the selected symbol in the Symbol 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 symbol in the Symbol 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 Symbol Layers

The basic workflow to associate graphic elements, custom properties, or named scripts to a symbol layer consists of these general steps:

- 1 Select a symbol 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 symbol 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 Symbol Layers

Graphic elements show the visual properties of a symbol. Designers must associate graphic elements to the defined layers of a symbol.

To associate graphic elements to symbol 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 symbol.
- 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 symbol layer.

You can also select the symbol element group by clicking it on the displayed symbol.

- 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 symbol 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 symbol.

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 symbol itself.

- 10 Click the **Show/Hide** icon to verify the graphic elements associated to a layer are correct for the symbol configuration.

- 11 Save your changes to the symbol.

Using Shortcut Menu Commands to Edit Symbol Layer Graphic Elements

The Symbol Wizard Editor provides a set of shortcut menu commands to add graphic elements to symbol 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 symbol layers.

To add graphic elements to symbol layers

- 1 Show the symbol with Symbol Wizard Editor selected.
- 2 From the **Layers** pane, select the check box next to symbol 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 symbol 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 Symbol Layers

Removing graphic elements from symbol 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 symbol layers

- 1 Show the symbol with Symbol Wizard Editor selected.

- 2 From the **Layers** pane, select the check box next to the symbol 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 removed 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 symbol 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 Symbol Layers

Associating custom properties to a symbol 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 symbol layer. You can associate custom properties to layers with the active layer or director folder methods.

To associate custom properties to symbol layers

- 1 Open the selected symbol in the Symbol Editor with the Symbol Wizard selected.
- 2 Click the **Custom Properties** tab to show the locally defined custom properties of the symbol.

Custom properties of embedded symbols 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 symbol 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 symbol layer.
- 9 Repeat steps 4-7 to select the remaining custom properties for the different layers of a symbol.
- 10 Save your changes to the symbol.

Associating Named Scripts to Symbol Layers

Associating named scripts to a symbol layer uses a similar procedure to associate graphic elements or custom properties. You can associate named scripts to layers with the active layer or director folder methods.

To associate named scripts to symbol layers

- 1 Show the selected symbol in the Symbol Editor with the Symbol Wizard selected.
- 2 Click the **Named Scripts** tab to show the scripts associated with the symbol.
- 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 symbol 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 symbol layer.
- 9 Repeat steps 4-7 to select the remaining scripts for the different layers of a symbol.
- 10 Save your changes to the symbol.

Verify Symbol Configurations

After creating the different configurations of a symbol, Designers use the Symbol Wizard Preview to verify each configuration works as designed. Also, Designers can validate the symbol to identify any invalid references to other objects or values.

To verify symbol configurations

- 1 Open the symbol created with Symbol Wizard in the Symbol Editor.
- 2 Click **Symbol Wizard Preview** shown on the menu bar of the Symbol Editor.

You can also open the Symbol Wizard Preview as a **View** menu option or by pressing Alt+P.

The Symbol Editor updates to show the **Wizard Options** view with a set of drop-down lists to select different symbol property attributes and options. The default symbol configuration should be selected.

- 3 Select the different combinations of property values and view options from **Wizard Options** fields.
- 4 Verify the symbol that appears is correct for the specified configuration Choices and Option rule.
- 5 Click the **Validation** icon to see if the symbol contains any invalid references.

The **Validation** view lists any invalid references within the symbol that need to be corrected.

Important: Invalid references also include references to properties or elements in hidden symbol layers.

Using Symbol Wizards in an Application

Symbol Wizards are stored in a Galaxy library just like standard ArcestrA Symbols. When a Consumer selects a symbol and embeds it into a managed InTouch application, the symbol's default configuration is selected.

The Consumer can change a Symbol Wizard's configuration by changing the values assigned to the symbol's properties from the Symbol Wizard's **Wizard Options** section of the **Properties** view. After selecting a symbol configuration and changing any properties, the Consumer saves the symbol.

The Symbol Wizard appears as the configuration selected by the Consumer. A Symbol Wizard's configuration cannot be changed during application run time.

Embedding Symbol Wizards

Consumers embed Symbol Wizards from the Graphic Toolbox. Embedding a Symbol Wizard is similar to embedding a standard ArcestrA symbol.

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 symbol

1 Create a new symbol from the Graphic Toolbox or add a symbol to a derived AutomationObject from the Template Toolbox.

2 Open the symbol to show the Symbol Editor.

3 On the **Edit** menu, click **Embed ArcestrA Graphic**.

You can also click the **Embed ArcestrA Graphic** icon from the menu bar.

The Galaxy Browser appears.

4 Locate the folder containing the Symbol Wizard.

5 Click the symbol 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 Symbol Editor canvas to show that it is selected.

8 Select the symbol's configuration by selecting values for the various options shown in the **Wizard Options** view.

9 Rename the symbol.

10 Right-click on the symbol 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 symbol's named scripts to run within the application.

14 Save the changes made to the symbol.

Configuring Symbol Wizards in WindowMaker

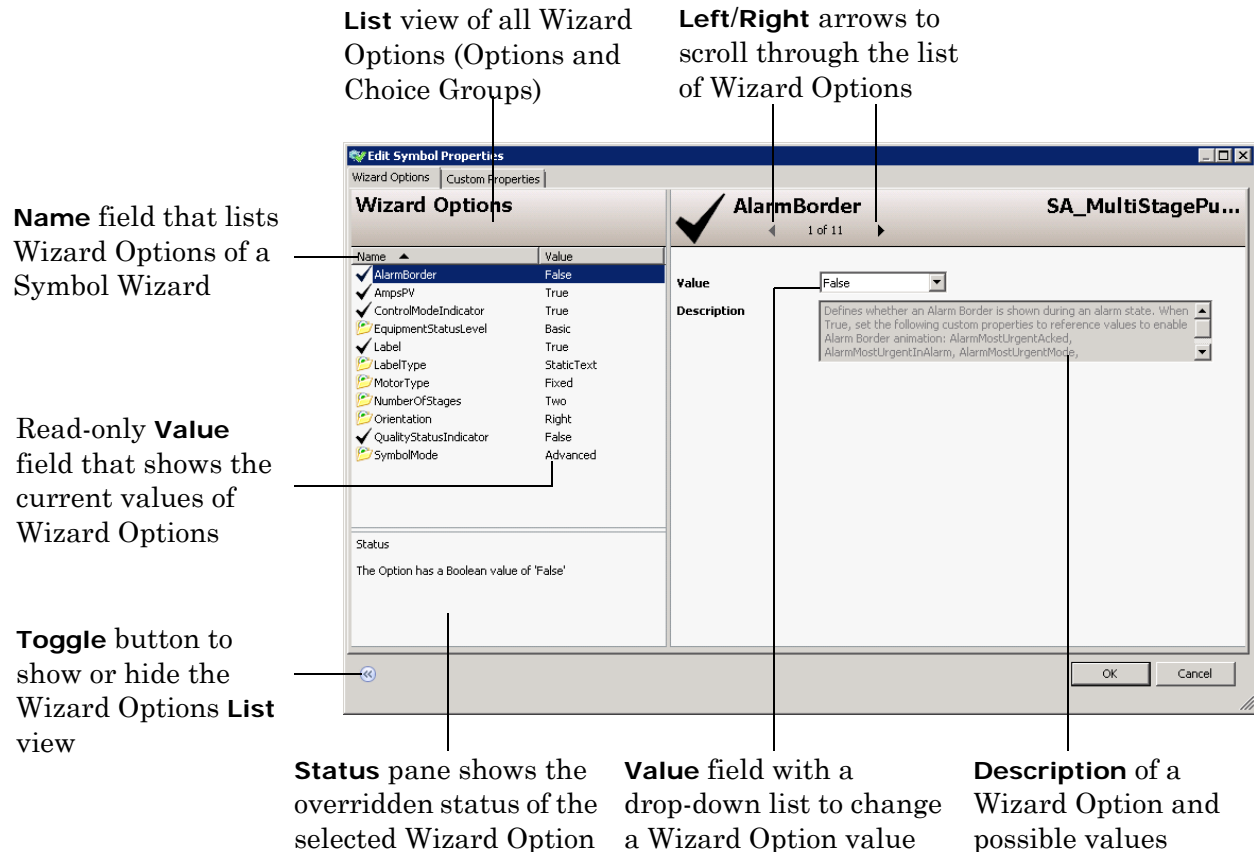
InTouch WindowMaker includes the ArcestrA Graphic Toolbox, which includes the entire set of ArcestrA and Situational Awareness Library symbols. Rather than embedding Symbol Wizards from the ArcestrA IDE, Symbol Wizards can be placed directly into an InTouch managed application window from WindowMaker.

Configuring a Symbol Wizard from WindowMaker is strictly the Consumer workflow of setting Wizard Options and custom properties to configure a symbol for an application. Designers must still work from the ArcestrA IDE to create Choice Groups, Choices, Layers, and rules to design Symbol Wizards.

The typical workflow to configure a Symbol Wizard from WindowMaker includes these major steps:

- 1 Place a Symbol Wizard in an InTouch window.
- 2 Open the Symbol Wizard's **Edit Symbol Properties** dialog box.
- 3 Set a Wizard Option value.
- 4 Set values to custom properties associated with the Wizard Option.
- 5 Repeat steps 3 and 4 until the Symbol Wizard is configured for the application.

The **Edit Symbol Properties** dialog box contains **Wizard Options** and **Custom Properties** tabs. You configure a Symbol Wizard from WindowMaker by setting values of the symbol's Wizard Options and custom properties. If a symbol does not contain any Wizard Options, the **Wizard Options** tab is hidden.



The **Value** field at the right contains a drop-down list of possible values that can be set for the selected Wizard Option. After a Wizard Option value is changed, both **Value** fields are dynamically updated. The appearance of the Symbol Wizard in the InTouch Window changes immediately with each update of a Wizard Option to reflect the current configuration. After changing a Wizard Option value, the list of custom properties also updates to show those custom properties that are associated with the symbol's selected configuration.

Many Situational Awareness Library symbols include a **SymbolMode** Wizard Option to show a **Basic** or **Advanced** set of Wizard Options. Select **Advanced** to see an expanded list of Wizard Options for a symbol. Select **Basic** to see a restricted list of Wizard Options and custom properties.

Clicking **OK** saves all updates of Wizard Options made from the **Edit Symbol Properties** dialog box.

To configure a Symbol Wizard in WindowMaker

- 1 Open a window of a managed InTouch application in WindowMaker.

Configuring a Symbol Wizard from WindowMaker can be done only for a managed InTouch application. Symbol Wizards cannot be configured for published or native InTouch applications in WindowMaker.

- 2 Select a Symbol Wizard from the ArcestrA Graphic Toolbox and drag and drop it onto the window.

The Symbol Wizard appears in the window and is selected.

- 3 Show the **Edit Symbol Properties** dialog box of the Symbol Wizard by one of the following methods:

- Double-click on the Symbol Wizard
- Press Ctrl + M
- Right-click to show the shortcut menu, select the symbol name from the list, and then select **Edit Symbol Properties** from the expanded window
- Select **Edit** from the menu bar, select the symbol name from the list, and then select **Edit Symbol Properties** from the expanded window

- 4 Select a Wizard Option from the List view that must be changed to configure the symbol for the InTouch application.

Both **Value** fields show the current assigned value of the Wizard Option.

- 5 Select a value from the drop-down list of the **Value** field at the right.
- 6 Select the **Custom Properties** tab to show the list of custom properties that can be modified for the current configuration of the Symbol Wizard.
- 7 Select a custom property from the list and change its value.
- 8 Repeat steps 4-7 until the Symbol Wizard is configured for the application.
- 9 Click **OK** to save your changes to the Symbol Wizard's Wizard Options and custom properties.

Symbol Wizard Tips and Examples

This section describes a practical example of creating a Symbol Wizard. The example explains how to modify an ArcestrA centrifugal pump symbol 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.

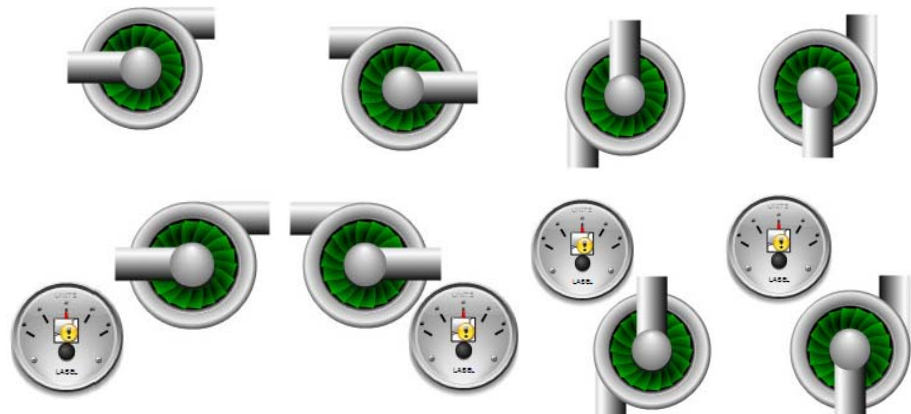
Creating Visual Configurations of an ArcestrA Symbol

Designers must complete the following tasks to create a Symbol Wizard:

- Plan the different configurations of a symbol and select a default configuration that represents the base Symbol Wizard.
- Identify the graphic elements needed to create each symbol configuration.
- Add graphic elements, named scripts, and custom properties for each configuration.
- Create symbol 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 HasTach=False |
|  | Orientation=Left HasTach=True |
|  | Orientation=Right HasTach=False |
|  | Orientation=Right HasTach=True |
|  | Orientation=Top HasTach=False |
|  | Orientation=Top HasTach=True |

| Symbol Configuration | Configuration Properties and Attributes |
|---|---|
|  | 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.

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 graphic elements, named scripts, and custom properties that you want to include in the base configuration.

Identify Symbol Elements

Symbol 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.

| Symbol Configuration | Configuration Properties and Attributes | Required Symbol Elements |
|---|---|---|
|  | Orientation=Left HasTach=False | Graphic elements: <ul style="list-style-type: none"> • InletLeft • OutletRight |
|  | Orientation=Left HasTach=True | Graphic elements: <ul style="list-style-type: none"> • InletLeft • OutletRight • MeterLeft |
|  | Orientation=Right HasTach=False | Graphic elements: <ul style="list-style-type: none"> • InletRight • OutleftLeft |
|  | Orientation=Right HasTach=True | Graphic elements: <ul style="list-style-type: none"> • InletRight • OutleftLeft • MeterRight |

| Symbol Configuration | Configuration Properties and Attributes | Required Symbol Elements |
|--|---|--|
|  | Orientation=Top HasTach=False | Graphic elements: <ul style="list-style-type: none"> • InletTop • OutletBottom |
|  | Orientation=Top HasTach=True | Graphic elements: <ul style="list-style-type: none"> • InletTop • OutletBottom • MeterTop |
|  | Orientation=Bottom HasTach=False | Graphic elements: <ul style="list-style-type: none"> • InletBottom • OutletTop |
|  | Orientation=Bottom HasTach=True | Graphic elements: <ul style="list-style-type: none"> • InletBottom • OutletTop • MeterTop |

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 symbol from the Graphic Toolbox in the ArcestrA Symbol Editor, or create a new symbol.
- 2 Create the graphic elements required for each Symbol Wizard configuration by doing one of the following:
 - Embed other symbols from the Graphic Toolbox into the symbol.
 - Duplicate graphic elements from the symbol and edit them as necessary for each Symbol Wizard configuration.



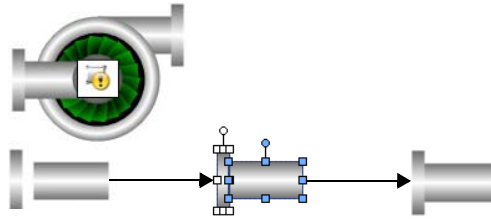
In the example of the Symbol Wizard centrifugal pump, an ArcestrA meter has been embedded into the symbol, and then duplicated for the different configuration positions. The inlet and outlet pipes are created by duplicating graphic pipe elements.

- 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

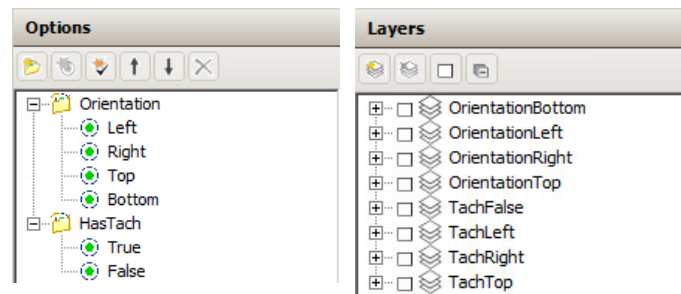
- If the same graphic element will be placed at different positions within a Symbol Wizard based on different configurations, create a copy of the graphic element for each position. Each graphic element can be placed into a separate layer, making it easier to specify rules to show the element at the 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 Symbol 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 symbol. Inlet and outlet graphic element pairs are added to each Orientation layer.



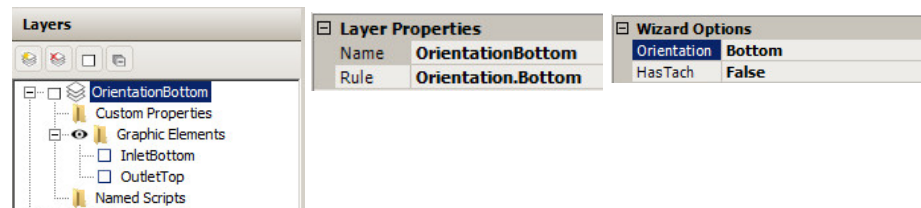
Layers need to be added for the left, right, and top positions of the tachometer when the HasTach Choice Group is True. Copies of the embedded tachometer symbol are added to the TachLeft, TachRight, and TachTop layers, which map to the different positions of the tachometer shown in the pump Symbol Wizard. 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 Symbol 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&HasTach.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&HasTach.True) | (Orientation.Bottom&HasTach.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"> <tr><td colspan="2">Wizard Options</td></tr> <tr><td>Orientation</td><td>Left</td></tr> <tr><td>HasTach</td><td>False</td></tr> </table> <p>Orientation.Left&HasTach.False</p> | Wizard Options | | Orientation | Left | HasTach | False |
| Wizard Options | | | | | | | |
| Orientation | Left | | | | | | |
| HasTach | False | | | | | | |
|  | <table border="1"> <tr><td colspan="2">Wizard Options</td></tr> <tr><td>Orientation</td><td>Left</td></tr> <tr><td>HasTach</td><td>True</td></tr> </table> <p>Orientation.Left&HasTach.True</p> | Wizard Options | | Orientation | Left | HasTach | True |
| Wizard Options | | | | | | | |
| Orientation | Left | | | | | | |
| HasTach | True | | | | | | |
|  | <table border="1"> <tr><td colspan="2">Wizard Options</td></tr> <tr><td>Orientation</td><td>Right</td></tr> <tr><td>HasTach</td><td>False</td></tr> </table> <p>Orientation.Right&HasTach.False</p> | Wizard Options | | Orientation | Right | HasTach | False |
| Wizard Options | | | | | | | |
| Orientation | Right | | | | | | |
| HasTach | False | | | | | | |
|  | <table border="1"> <tr><td colspan="2">Wizard Options</td></tr> <tr><td>Orientation</td><td>Right</td></tr> <tr><td>HasTach</td><td>True</td></tr> </table> <p>Orientation.Right&HasTach.True</p> | Wizard Options | | Orientation | Right | HasTach | True |
| Wizard Options | | | | | | | |
| Orientation | Right | | | | | | |
| HasTach | True | | | | | | |
|  | <table border="1"> <tr><td colspan="2">Wizard Options</td></tr> <tr><td>Orientation</td><td>Top</td></tr> <tr><td>HasTach</td><td>False</td></tr> </table> <p>Orientation.Top&HasTach.False</p> | Wizard Options | | Orientation | Top | HasTach | False |
| Wizard Options | | | | | | | |
| Orientation | Top | | | | | | |
| HasTach | False | | | | | | |
|  | <table border="1"> <tr><td colspan="2">Wizard Options</td></tr> <tr><td>Orientation</td><td>Top</td></tr> <tr><td>HasTach</td><td>True</td></tr> </table> <p>(Orientation.Top&HasTach.True) (Orientation.Bottom&HasTach.True)</p> | Wizard Options | | Orientation | Top | HasTach | True |
| Wizard Options | | | | | | | |
| Orientation | Top | | | | | | |
| HasTach | True | | | | | | |

**Symbol
Wizard
Configuration**
**Wizard Options and Corresponding Active
Configuration Layer Rules**



| 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

This section shows you the properties of elements, the canvas, element groups, and embedded symbols.

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.

Alphabetical List of Properties

The following table contains a list of properties used by the:

- Elements.
- Canvas.
- Element groups.
- Embedded symbols.

Asterisk (*) marks properties that are specific to only one type of element or the canvas, a group or an embedded symbol.

| Property | Purpose, category, usage and further information |
|-----------------|--|
| AbsoluteAnchor* | <p>Purpose: Defines the absolute anchor point of the source symbol. By default, this is the center point of all elements on the canvas but can be changed.</p> <p>Category: Appearance</p> <p>Used by: Canvas</p> <p>Can be read by script at run time: No</p> <p>Info: "Size Propagation and Anchor Points" on page 43</p> |
| AbsoluteOrigin | <p>Purpose: Defines an X, Y location relative to the top, left (0, 0) origin of the symbol or window.</p> <p>Category: Appearance</p> <p>Used by: 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>Can be read by script at run time: No</p> <p>Info: "Changing Points of Origin in the Properties Editor" on page 134</p> |
| Alignment | <p>Purpose: Controls the location of the text relative to the bounding rectangle of the element.</p> <p>Category: Text Style</p> <p>Used by: Button, Text, Text Box</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Text Alignment" on page 173</p> |
| AnchorFixedTo | <p>Purpose: 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>Category: Appearance</p> <p>Used by: Embedded Symbol, Canvas</p> <p>Can be read by script at run time: No</p> <p>Info: "Size Propagation and Anchor Points" on page 43</p> |

| Property | Purpose, category, usage and further information |
|--------------|--|
| AnchorPoint* | <p>Purpose: Defines the anchor X, Y location of the embedded symbol.</p> <p>Category: Appearance</p> <p>Used by: Embedded Symbol</p> <p>Can be read by script at run time: No</p> <p>Info: "Size Propagation and Anchor Points" on page 43</p> |
| Angle | <p>Purpose: 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>Category: Appearance</p> <p>Used by: 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>Can be read by script at run time: Yes</p> <p>Info: "Rotating Elements by Changing the Angle Property" on page 133</p> |
| AutoScale | <p>Purpose: 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>Category: Appearance</p> <p>Used by: Button, Text Box</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Setting Auto Scaling and Word Wrapping for a Text Box" on page 212</p> |
| ButtonStyle* | <p>Purpose: Determines if the button appears as a standard button or as an image.</p> <p>Category: Appearance</p> <p>Used by: Button</p> <p>Can be read by script at run time: No</p> <p>Info: "Configuring Buttons with Images" on page 217</p> |

| Property | Purpose, category, usage and further information |
|------------------|--|
| CalendarColumns* | <p>Purpose: Defines the number of columns the calendar object has.</p> <p>Category: Appearance</p> <p>Used by: Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Number of Calendar Month Sheets" on page 247</p> |
| CalendarRows* | <p>Purpose: Defines the number of rows the calendar object has.</p> <p>Category: Appearance</p> <p>Used by: Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Number of Calendar Month Sheets" on page 247</p> |
| Caption* | <p>Purpose: 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>Category: Text Style</p> <p>Used by: Check Box</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Caption Text of a Check Box Control" on page 242</p> |
| Checked* | <p>Purpose: 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>Category: Appearance</p> <p>Used by: Check Box</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Setting the Default State of a Check Box Control" on page 242</p> |

| Property | Purpose, category, usage and further information |
|--------------|--|
| .Color1 | <p>Purpose: 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>Category: Depends on its source property</p> <p>Used by: Depends on its source property</p> <p>Can be read by script at run time: No</p> <p>Info: "Enabling and Disabling Elements for Run-Time Interaction" on page 188</p> |
| .Color2 | <p>Purpose: 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>Category: Depends on its source property</p> <p>Used by: Depends on its source property</p> <p>Can be read by script at run time: No</p> <p>Info: "Enabling and Disabling Elements for Run-Time Interaction" on page 188</p> |
| .Color3 | <p>Purpose: 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>Category: Depends on its source property</p> <p>Used by: Depends on its source property</p> <p>Can be read by script at run time: No</p> <p>Info: "Enabling and Disabling Elements for Run-Time Interaction" on page 188</p> |
| ControlStyle | <p>Purpose: Defines the control style as Flat or 3D.</p> <p>Category: Appearance</p> <p>Used by: Radio Button Group, Check Box</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Layout of the Radio Button Group Options" on page 241 and "Setting the 3D appearance of a Check Box Control" on page 243</p> |

| Property | Purpose, category, usage and further information |
|------------------|--|
| Count | <p>Purpose: Indicates how many items there are in a list.</p> <p>Category: not available at design time</p> <p>Used by: Radio Button Group, Combo Box, List Box</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Using Radio Button Group-Specific Properties at Run Time" on page 241, "Using Combo Box-Specific Properties at Run Time" on page 246 and "Using List Box-Specific Properties at Run Time" on page 252</p> |
| CustomFormat* | <p>Purpose: Defines the format to be used in the DateTime Picker control for input of a date or time.</p> <p>Category: Appearance</p> <p>Used by: DateTime Picker</p> <p>Can be read by script at run time: No</p> <p>Info: "Configuring DateTime Picker Controls" on page 249</p> |
| CustomProperties | <p>Purpose: The collection of CustomProperties defined by the symbol.</p> <p>Category: Custom Properties</p> <p>Used by: Canvas, Embedded Symbol</p> <p>Can be read by script at run time: No</p> <p>Info: "Using Custom Properties" on page 253</p> |
| Description* | <p>Purpose: Contains a meaningful description of the symbol.</p> <p>Category: Graphic</p> <p>Used by: Canvas</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Radius of Rounded Rectangles" on page 210</p> |
| DefaultValue | <p>Purpose: The default time value to use for the control.</p> <p>Category: Appearance</p> <p>Used by: Calendar, DateTime Picker</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Default Value of the Calendar Control" on page 249 and "Configuring DateTime Picker Controls" on page 249</p> |

| Property | Purpose, category, usage and further information |
|--------------------|--|
| DownImage* | <p>Purpose: Defines the image that is rendered in the button element when it is clicked or held down.</p> <p>Category: Appearance</p> <p>Used by: Button</p> <p>Can be read by script at run time: No</p> <p>Info: "Configuring Buttons with Images" on page 217</p> |
| DropDownType* | <p>Purpose: Defines the type of combo box: simple, drop-down or drop-down list.</p> <p>Category: Appearance</p> <p>Used by: Combo Box</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Type of Combo Box Control" on page 244</p> |
| DropDownWidth* | <p>Purpose: Defines the width of the drop-down list.</p> <p>Category: Appearance</p> <p>Used by: Combo Box</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Width of the Drop-Down List" on page 245</p> |
| DynamicSizeChange* | <p>Purpose: Determines if the embedded symbol propagates the size changes from the source symbol.</p> <p>Category: Appearance</p> <p>Used by: Embedded Symbol</p> <p>Can be read by script at run time: No</p> <p>Info: "Enabling or Disabling Dynamic Size Change of Embedded Symbols" on page 413</p> |

| Property | Purpose, category, usage and further information |
|--------------|--|
| Enabled | <p>Purpose: 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>Category: Runtime Behavior</p> <p>Used by: 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>Can be read by script at run time: Yes</p> <p>Info: "Enabling and Disabling Elements for Run-Time Interaction" on page 188</p> |
| End | <p>Purpose: Defines the end of a line or H/V line as X, Y location.</p> <p>Category: Appearance</p> <p>Used by: Line, H/V Line</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting Start or End Points of a Line" on page 169</p> |
| EndCap | <p>Purpose: Defines the cap used at the end of the line of an open element.</p> <p>Category: Line Style</p> <p>Used by: Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting Line End Shape and Size" on page 211</p> |
| FillBehavior | <p>Purpose: Determines how the Fill (Horizontal, Vertical or Both) should be applied to the element.</p> <p>Category: Fill Style</p> <p>Used by: 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>Can be read by script at run time: No</p> <p>Info: "Setting Fill Behavior" on page 167</p> |

| Property | Purpose, category, usage and further information |
|-----------------|---|
| FillColor | <p>Purpose: Defines the fill style used for the filled portion of the element.</p> <p>Category: Fill Style</p> <p>Used by: 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>Can be read by script at run time: No</p> <p>Info: "Setting Fill Style" on page 165 and "Changing Background Color and Text Color of Windows Common Controls" on page 239</p> |
| FillOrientation | <p>Purpose: Determines the orientation of the fill when the element orientation is any value other than 0.</p> <p>Category: Fill Style</p> <p>Used by: 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>Can be read by script at run time: No</p> <p>Info: "Setting Fill Orientation" on page 166</p> |
| FirstDayOfWeek* | <p>Purpose: Defines the first day of the week used for the display of the columns in the calendar.</p> <p>Category: Appearance</p> <p>Used by: Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the First Day of the Week" on page 247</p> |
| Font | <p>Purpose: Defines the basic text font as defined by the operating system.</p> <p>Category: Text Style</p> <p>Used by: Button, Text, Text Box, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Text Font" on page 172</p> |

| Property | Purpose, category, usage and further information |
|----------------------|--|
| Format* | <p>Purpose: Defines the format of the reference values. This is only available for array mode.</p> <p>Category: Appearance</p> <p>Used by: DateTime Picker</p> <p>Can be read by script at run time: No</p> <p>Info: "Configuring DateTime Picker Controls" on page 249</p> |
| HasTransparentColor* | <p>Purpose: 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>Category: Appearance</p> <p>Used by: Image</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Setting the Image Color Transparency" on page 214</p> |
| Height | <p>Purpose: Defines the height of the element.</p> <p>Category: Appearance</p> <p>Used by: 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>Can be read by script at run time: Yes</p> <p>Info: "Resizing Elements by Changing Size Properties" on page 129</p> |
| HorizontalDirection | <p>Purpose: Determines the horizontal direction of the fill for the element. Can be "Right" or "Left".</p> <p>Category: Fill Style</p> <p>Used by: 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>Can be read by script at run time: No</p> <p>Info: "Setting Horizontal Fill Direction and Percentage" on page 167</p> |

| Property | Purpose, category, usage and further information |
|-----------------------|---|
| HorizontalPercentFill | <p>Purpose: Determines the percentage of horizontal fill for the element.</p> <p>Category: Fill Style</p> <p>Used by: 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>Can be read by script at run time: Yes</p> <p>Info: "Setting Horizontal Fill Direction and Percentage" on page 167</p> |
| HorizontalScrollbar | <p>Purpose: 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>Category: Appearance</p> <p>Used by: List Box</p> <p>Can be read by script at run time: No</p> <p>Info: "Using a Horizontal Scroll Bar in a List Box Control" on page 252</p> |
| Image* | <p>Purpose: Defines the image that is rendered in the element. Any image format supported by the application can be used.</p> <p>Category: Appearance</p> <p>Used by: Image</p> <p>Can be read by script at run time: No</p> <p>Info: "Selecting a Different Image" on page 216</p> |
| ImageAlignment* | <p>Purpose: 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>Category: Appearance</p> <p>Used by: Image</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Image Alignment" on page 214</p> |

| Property | Purpose, category, usage and further information |
|----------------|--|
| ImageStyle | <p>Purpose: Defines how the image is rendered relative to its bounding rectangle.</p> <p>Category: Appearance</p> <p>Used by: Button, Image</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Image Display Mode" on page 213</p> |
| IntegralHeight | <p>Purpose: 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>Category: Appearance</p> <p>Used by: Combo Box, List Box</p> <p>Can be read by script at run time: No</p> <p>Info: "Avoiding Clipping of Items in the Simple Combo Box Control" on page 245</p> |
| Language | <p>Purpose: Defines the current language of the symbol.</p> <p>Category: Runtime Behavior</p> <p>Used by: Embedded Symbol</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Selecting the Language for a Symbol" on page 428.</p> |
| LanguageID | <p>Purpose: Defines the current language ID of the symbol.</p> <p>Category: Runtime Behavior</p> <p>Used by: Embedded Symbol</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Selecting the Language for a Symbol" on page 428.</p> |
| Layout* | <p>Purpose: Defines the way the radio buttons are arranged in the group (Horizontal or Vertical).</p> <p>Category: Appearance</p> <p>Used by: Radio Button Group</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Layout of the Radio Button Group Options" on page 241</p> |

| Property | Purpose, category, usage and further information |
|-------------|---|
| LineColor | <p>Purpose: Defines the color and affects of the line or border.</p> <p>Category: Line Style</p> <p>Used by: 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>Can be read by script at run time: No</p> <p>Info: "Setting the Line Style" on page 170</p> |
| LinePattern | <p>Purpose: Defines the pattern of the line or border.</p> <p>Category: Line Style</p> <p>Used by: 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>Can be read by script at run time: No</p> <p>Info: "Setting the Line Pattern" on page 169</p> |
| LineWeight | <p>Purpose: Determines the weight of the element's line or border. A value of 0 means that there is no line or border.</p> <p>Category: Line Style</p> <p>Used by: 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>Can be read by script at run time: Yes</p> <p>Info: "Setting the Line Weight" on page 169</p> |
| Locked | <p>Purpose: 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>Category: Appearance</p> <p>Used by: 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>Can be read by script at run time: Yes</p> <p>Info: "Locking and Unlocking Elements" on page 148</p> |

| Property | Purpose, category, usage and further information |
|------------------------|---|
| MaxDropDownItems* | <p>Purpose: Defines the maximum number of items the drop-down list shows.</p> <p>Category: Appearance</p> <p>Used by: Combo Box</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Maximum Number of Items to Appear in the Combo Box Drop-Down List" on page 246</p> |
| Multiline* | <p>Purpose: Determines if the control shows several lines of text that automatically wrap up when reaching the right border of the control.</p> <p>Category: Appearance</p> <p>Used by: Edit Box</p> <p>Can be read by script at run time: No</p> <p>Info: "Configuring the Text to Wrap in an Edit Box Control" on page 243</p> |
| MultiplePopupsAllowed* | <p>Purpose: 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>Category: Runtime Behavior</p> <p>Used by: Canvas</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Radius of Rounded Rectangles" on page 210</p> |
| Name | <p>Purpose: Gives the element a meaningful unique name.</p> <p>Category: Graphic</p> <p>Used by: 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>Can be read by script at run time: No</p> <p>Info: "Setting the Radius of Rounded Rectangles" on page 210</p> |

| Property | Purpose, category, usage and further information |
|---------------|---|
| NewIndex | <p>Purpose: Returns the index of the last value added to the list. This is provided for migration of InTouch Windows common controls.</p> <p>Category: not available at design time</p> <p>Used by: Combo Box, List Box</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Using Combo Box-Specific Properties at Run Time" on page 246 and "Using List Box-Specific Properties at Run Time" on page 252</p> |
| OwningObject* | <p>Purpose: Used as the ArcestraA 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>Category: Runtime Behavior</p> <p>Used by: Embedded Symbol</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Detecting and Editing the Containing AutomationObject Instance" on page 415</p> |
| Radius* | <p>Purpose: Defines the radius of the corners of the Rounded Rectangle.</p> <p>Category: Appearance</p> <p>Used by: Rounded Rectangle</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Setting the Radius of Rounded Rectangles" on page 210</p> |
| ReadOnly* | <p>Purpose: Determines if the user can type data into the edit box.</p> <p>Category: Appearance</p> <p>Used by: Edit Box</p> <p>Can be read by script at run time: No</p> <p>Info: "Configuring the Text to be Read-Only in an Edit Box Control" on page 244</p> |

| Property | Purpose, category, usage and further information |
|-----------------|--|
| RelativeAnchor* | <p>Purpose: Relative anchor point of the source symbol. By default, this is 0,0.</p> <p>Category: Appearance</p> <p>Used by: Canvas</p> <p>Can be read by script at run time: No</p> <p>Info: "Size Propagation and Anchor Points" on page 43</p> |
| RelativeOrigin | <p>Purpose: Defines the relative origin as X, Y location. The location is relative to the center point of the element (0, 0).</p> <p>Category: Appearance</p> <p>Used by: 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>Can be read by script at run time: No</p> <p>Info: "Changing Points of Origin in the Properties Editor" on page 134</p> |
| Scripts* | <p>Purpose: Defines a collection of scripts configured for the symbol.</p> <p>Category: Runtime Behavior</p> <p>Used by: Canvas</p> <p>Can be read by script at run time: No</p> <p>Info: "Adding and Maintaining Symbol Scripts" on page 369</p> |
| SelectedValue | <p>Purpose: Reads the value of the selected item, or selects the item with that value if it exists.</p> <p>Category: not available at design time</p> <p>Used by: Radio Button Group, List Box</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Using Radio Button Group-Specific Properties at Run Time" on page 241 and "Using List Box-Specific Properties at Run Time" on page 252</p> |

| Property | Purpose, category, usage and further information |
|------------|---|
| ShowToday* | <p>Purpose: Determines if today's date is shown on the calendar control.</p> <p>Category: Appearance</p> <p>Used by: Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: "Showing or Hiding Today's Date on a Calendar Control" on page 248</p> |
| Smoothing* | <p>Purpose: When False the graphics are rendered normally, when True graphics are rendered with anti-aliasing which produces a smoother appearing graphic.</p> <p>Category: Appearance</p> <p>Used by: Canvas</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Radius of Rounded Rectangles" on page 210</p> |
| Start | <p>Purpose: Defines the start of a line or H/V line as X, Y location.</p> <p>Category: Appearance</p> <p>Used by: Line, H/V Line</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting Start or End Points of a Line" on page 169</p> |
| StartAngle | <p>Purpose: 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>Category: Appearance</p> <p>Used by: 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, 2 Point Arc, 3 Point Arc</p> <p>Can be read by script at run time: No</p> <p>Info: "Changing Angles of Arcs, Pies and Chords" on page 219</p> |

| Property | Purpose, category, usage and further information |
|------------------|---|
| StartCap | <p>Purpose: Defines the cap used at the start of the line of an open graphic.</p> <p>Category: Line Style</p> <p>Used by: Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting Line End Shape and Size" on page 211</p> |
| SweepAngle | <p>Purpose: Defines the ending angle of the Arc, Pie or Chord. This angle is always measured from the start angle.</p> <p>Category: Appearance</p> <p>Used by: 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, 2 Point Arc, 3 Point Arc</p> <p>Can be read by script at run time: No</p> <p>Info: "Changing Angles of Arcs, Pies and Chords" on page 219</p> |
| SymbolReference* | <p>Purpose: 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> <hr/> <p>Note: This property is always disabled.</p> <hr/> <p>Category: Runtime Behavior</p> <p>Used by: Embedded Symbol</p> <p>Can be read by script at run time: No</p> <p>Info: "Detecting the Source Symbol of an Embedded Symbol" on page 411</p> |
| TabOrder | <p>Purpose: 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>Category: Runtime Behavior</p> <p>Used by: 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>Can be read by script at run time: No</p> <p>Info: "Editing the Tab Order of an Element" on page 189</p> |

| Property | Purpose, category, usage and further information |
|-----------|---|
| TabStop | <p>Purpose: Determines if the element can be navigated to and can receive focus at run time.</p> <p>Category: Runtime Behavior</p> <p>Used by: 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>Can be read by script at run time: No</p> <p>Info: "Editing the Tab Order of an Element" on page 189</p> |
| Tension | <p>Purpose: Specifies how tightly the curve bends through the control points of the curve.</p> <p>Category: Appearance</p> <p>Used by: Closed Curve, Curve</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Changing the Tension of Curves and Closed Curves" on page 219</p> |
| Text | <p>Purpose: Defines the unicode text that is shown by the element.</p> <p>Category: Appearance</p> <p>Used by: Button, Text, Text Box, Edit Box</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Setting the Displayed Text" on page 171</p> |
| TextColor | <p>Purpose: Defines the color and affects applied to the text.</p> <p>Category: Text Style</p> <p>Used by: Button, Text, Text Box, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Text Color" on page 172 and "Changing Background Color and Text Color of Windows Common Controls" on page 239</p> |

| Property | Purpose, category, usage and further information |
|-----------------|--|
| TextFormat | <p>Purpose: Defines the formatting string that is applied to the text when it is shown.</p> <p>Category: Appearance</p> <p>Used by: Button, Text, Text Box</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Setting the Text Display Format" on page 171</p> |
| TitleFillColor* | <p>Purpose: Determines the background solid color in the title bar of the calendar control.</p> <p>Category: Fill Style</p> <p>Used by: Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting Title Fill Color and Text Color on a Calendar Control" on page 248</p> |
| TitleTextColor* | <p>Purpose: Determines the text solid color in the title bar of the calendar control.</p> <p>Category: Text Style</p> <p>Used by: Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting Title Fill Color and Text Color on a Calendar Control" on page 248</p> |
| TopIndex* | <p>Purpose: Returns the index of the top most item in the list. This is provided for migration of InTouch Windows common controls.</p> <p>Category: not available at design time</p> <p>Used by: List Box</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Using List Box-Specific Properties at Run Time" on page 252</p> |

| Property | Purpose, category, usage and further information |
|--------------------|--|
| TrailingTextColor* | <p>Purpose: Determines the text solid color of the text for the trailing days. The trailing days are days outside the current month.</p> <p>Category: Text Style</p> <p>Used by: Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Text Color for Trailing Dates in a Calendar Control" on page 249</p> |
| Transparency | <p>Purpose: Defines the transparency of the element. A value of 0 means fully opaque and a value of 100 means fully transparent.</p> <p>Category: Appearance</p> <p>Used by: 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>Can be read by script at run time: Yes</p> <p>Info: "Setting the Transparency Level of an Element" on page 186</p> |
| .Transparency | <p>Purpose: 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>Category: Depends on its source property</p> <p>Used by: Depends on its source property</p> <p>Can be read by script at run time: No</p> <p>Info: "Enabling and Disabling Elements for Run-Time Interaction" on page 188</p> |
| TransparentColor* | <p>Purpose: Defines the RGB color value that is used as the transparent color.</p> <p>Category: Appearance</p> <p>Used by: Image</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Image Color Transparency" on page 214</p> |

| Property | Purpose, category, usage and further information |
|---------------|--|
| TreatAsIcon | <p>Purpose: If this property is set to False, the animations defined on the graphics within the group or embedded symbol take precedence over an animation defined on the group or embedded symbol. If there are no animations or the user clicked on an area of the group or embedded symbol that does not have an animation, then the group or embedded symbol animation executes.</p> <p>If the property is set to True, only the animation on the group or embedded symbol is executed. The interactive animations within the group or embedded symbol never execute.</p> <p>Category: Runtime Behavior</p> <p>Used by: Group, Embedded Symbol</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Editing the Embedded Symbol" on page 408</p> |
| UnFilledColor | <p>Purpose: Determines the element's unfilled area appearance.</p> <p>Category: Fill Style</p> <p>Used by: 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>Can be read by script at run time: No</p> <p>Info: "Setting Unfilled Style" on page 166</p> |
| UpImage* | <p>Purpose: Defines the image that is used in the button element when it is un-clicked or released.</p> <p>Category: Appearance</p> <p>Used by: Button</p> <p>Can be read by script at run time: No</p> <p>Info: "Configuring Buttons with Images" on page 217</p> |
| Value | <p>Purpose: 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>Category: not available at design time</p> <p>Used by: Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Reading and Writing the Selected Value at Run Time" on page 239</p> |

| Property | Purpose, category, usage and further information |
|---------------------|--|
| VerticalDirection | <p>Purpose: Defines the vertical direction of the fill. Can be "Top" or "Bottom".</p> <p>Category: Fill Style</p> <p>Used by: 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>Can be read by script at run time: No</p> <p>Info: "Setting Vertical Fill Direction and Percentage" on page 168</p> |
| VerticalPercentFill | <p>Purpose: Determines the percentage of vertical fill for the element.</p> <p>Category: Fill Style</p> <p>Used by: 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>Can be read by script at run time: Yes</p> <p>Info: "Setting Vertical Fill Direction and Percentage" on page 168</p> |
| Visible | <p>Purpose: 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>Category: Runtime Behavior</p> <p>Used by: 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>Can be read by script at run time: Yes</p> <p>Info: "Changing the Visibility of Elements" on page 189</p> |

| Property | Purpose, category, usage and further information |
|----------|---|
| Width | <p>Purpose: Defines the width of the element.</p> <p>Category: Appearance</p> <p>Used by: 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>Can be read by script at run time: Yes</p> <p>Info: "Resizing Elements by Changing Size Properties" on page 129</p> |
| WordWrap | <p>Purpose: 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>Category: Appearance</p> <p>Used by: Button, Text Box</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Wrapping Text in Buttons" on page 217</p> |

| Property | Purpose, category, usage and further information |
|----------|---|
| X | <p>Purpose: Defines the left position of the element.</p> <p>Category: Appearance</p> <p>Used by: 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>Can be read by script at run time: Yes</p> <p>Info: "Moving Elements" on page 121</p> |
| Y | <p>Purpose: Defines the top position of the element.</p> <p>Category: Appearance</p> <p>Used by: 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>Can be read by script at run time: Yes</p> <p>Info: "Moving Elements" on page 121</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 symbols.

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 symbol.

| Property | Purpose, category, usage and further information |
|--------------|---|
| Description* | <p>Purpose: Contains a meaningful description of the symbol.</p> <p>Category: Graphic</p> <p>Used by: Canvas</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Radius of Rounded Rectangles" on page 210</p> |
| Name | <p>Purpose: Gives the element a meaningful unique name.</p> <p>Category: Graphic</p> <p>Used by: 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>Can be read by script at run time: No</p> <p>Info: "Setting the Radius of Rounded Rectangles" on page 210</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 symbols.

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 symbol.

| Property | Purpose, category, usage and further information |
|-----------------|--|
| AbsoluteAnchor* | <p>Purpose: Defines the absolute anchor point of the source symbol. By default, this is the center point of all elements on the canvas but can be changed.</p> <p>Category: Appearance</p> <p>Used by: Canvas</p> <p>Can be read by script at run time: No</p> <p>Info: "Size Propagation and Anchor Points" on page 43</p> |
| AnchorFixedTo | <p>Purpose: 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>Category: Appearance</p> <p>Used by: Embedded Symbol, Canvas</p> <p>Can be read by script at run time: No</p> <p>Info: "Size Propagation and Anchor Points" on page 43</p> |
| AbsoluteOrigin | <p>Purpose: Defines an X, Y location relative to the top, left (0, 0) origin of the symbol or window.</p> <p>Category: Appearance</p> <p>Used by: 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>Can be read by script at run time: No</p> <p>Info: "Changing Points of Origin in the Properties Editor" on page 134</p> |
| AnchorPoint* | <p>Purpose: Defines the anchor X, Y location of the embedded symbol.</p> <p>Category: Appearance</p> <p>Used by: Embedded Symbol</p> <p>Can be read by script at run time: No</p> <p>Info: "Size Propagation and Anchor Points" on page 43</p> |

| Property | Purpose, category, usage and further information |
|------------------|--|
| Angle | <p>Purpose: 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>Category: Appearance</p> <p>Used by: 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>Can be read by script at run time: Yes</p> <p>Info: "Rotating Elements by Changing the Angle Property" on page 133</p> |
| AutoScale | <p>Purpose: 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>Category: Appearance</p> <p>Used by: Button, Text Box</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Setting Auto Scaling and Word Wrapping for a Text Box" on page 212</p> |
| ButtonStyle* | <p>Purpose: Determines if the button appears as a standard button or as an image.</p> <p>Category: Appearance</p> <p>Used by: Button</p> <p>Can be read by script at run time: No</p> <p>Info: "Configuring Buttons with Images" on page 217</p> |
| CalendarColumns* | <p>Purpose: Defines the number of columns the calendar object has.</p> <p>Category: Appearance</p> <p>Used by: Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Number of Calendar Month Sheets" on page 247</p> |

| Property | Purpose, category, usage and further information |
|---------------|--|
| CalendarRows* | <p>Purpose: Defines the number of rows the calendar object has.</p> <p>Category: Appearance</p> <p>Used by: Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Number of Calendar Month Sheets" on page 247</p> |
| Checked* | <p>Purpose: 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>Category: Appearance</p> <p>Used by: Check Box</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Setting the Default State of a Check Box Control" on page 242</p> |
| ControlStyle | <p>Purpose: Defines the control style as Flat or 3D.</p> <p>Category: Appearance</p> <p>Used by: Radio Button Group, Check Box</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Layout of the Radio Button Group Options" on page 241 and "Setting the 3D appearance of a Check Box Control" on page 243</p> |
| CustomFormat* | <p>Purpose: Defines the format to be used in the DateTime Picker control for input of a date or time.</p> <p>Category: Appearance</p> <p>Used by: DateTime Picker</p> <p>Can be read by script at run time: No</p> <p>Info: "Configuring DateTime Picker Controls" on page 249</p> |
| DefaultValue | <p>Purpose: The default time value to use for the control.</p> <p>Category: Appearance</p> <p>Used by: Calendar, DateTime Picker</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Default Value of the Calendar Control" on page 249 and "Configuring DateTime Picker Controls" on page 249</p> |

| Property | Purpose, category, usage and further information |
|--------------------|--|
| DownImage* | <p>Purpose: Defines the image that is rendered in the button element when it is clicked or held down.</p> <p>Category: Appearance</p> <p>Used by: Button</p> <p>Can be read by script at run time: No</p> <p>Info: "Configuring Buttons with Images" on page 217</p> |
| DropDownType* | <p>Purpose: Defines the type of combo box: simple, drop-down or drop-down list.</p> <p>Category: Appearance</p> <p>Used by: Combo Box</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Type of Combo Box Control" on page 244</p> |
| DropDownWidth* | <p>Purpose: Defines the width of the drop-down list.</p> <p>Category: Appearance</p> <p>Used by: Combo Box</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Width of the Drop-Down List" on page 245</p> |
| DynamicSizeChange* | <p>Purpose: Determines if the embedded symbol propagates the size changes from the source symbol.</p> <p>Category: Appearance</p> <p>Used by: Embedded Symbol</p> <p>Can be read by script at run time: No</p> <p>Info: "Enabling or Disabling Dynamic Size Change of Embedded Symbols" on page 413</p> |
| End | <p>Purpose: Defines the end of a line or H/V line as X, Y location.</p> <p>Category: Appearance</p> <p>Used by: Line, H/V Line</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting Start or End Points of a Line" on page 169</p> |

| Property | Purpose, category, usage and further information |
|----------------------|--|
| FirstDayOfWeek* | <p>Purpose: Defines the first day of the week used for the display of the columns in the calendar.</p> <p>Category: Appearance</p> <p>Used by: Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the First Day of the Week" on page 247</p> |
| Format* | <p>Purpose: Defines the format of the reference values. This is only available for array mode.</p> <p>Category: Appearance</p> <p>Used by: DateTime Picker</p> <p>Can be read by script at run time: No</p> <p>Info: "Configuring DateTime Picker Controls" on page 249</p> |
| HasTransparentColor* | <p>Purpose: 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>Category: Appearance</p> <p>Used by: Image</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Setting the Image Color Transparency" on page 214</p> |
| Height | <p>Purpose: Defines the height of the element.</p> <p>Category: Appearance</p> <p>Used by: 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>Can be read by script at run time: Yes</p> <p>Info: "Resizing Elements by Changing Size Properties" on page 129</p> |

| Property | Purpose, category, usage and further information |
|---------------------|---|
| HorizontalScrollbar | <p data-bbox="581 268 1390 365">Purpose: 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 data-bbox="581 390 878 420">Category: Appearance</p> <p data-bbox="581 445 821 474">Used by: List Box</p> <p data-bbox="581 499 1114 529">Can be read by script at run time: No</p> <p data-bbox="581 554 1377 609">Info: "Using a Horizontal Scroll Bar in a List Box Control" on page 252</p> |
| Image* | <p data-bbox="581 630 1365 697">Purpose: Defines the image that is rendered in the element. Any image format supported by the application can be used.</p> <p data-bbox="581 722 878 751">Category: Appearance</p> <p data-bbox="581 777 792 806">Used by: Image</p> <p data-bbox="581 831 1114 861">Can be read by script at run time: No</p> <p data-bbox="581 886 1195 915">Info: "Selecting a Different Image" on page 216</p> |
| ImageAlignment* | <p data-bbox="581 936 1341 1024">Purpose: 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 data-bbox="581 1050 878 1079">Category: Appearance</p> <p data-bbox="581 1104 792 1134">Used by: Image</p> <p data-bbox="581 1159 1114 1188">Can be read by script at run time: No</p> <p data-bbox="581 1213 1214 1243">Info: "Setting the Image Alignment" on page 214</p> |
| ImageStyle | <p data-bbox="581 1264 1341 1331">Purpose: Defines how the image is rendered relative to its bounding rectangle.</p> <p data-bbox="581 1356 878 1386">Category: Appearance</p> <p data-bbox="581 1411 894 1440">Used by: Button, Image</p> <p data-bbox="581 1465 1114 1495">Can be read by script at run time: No</p> <p data-bbox="581 1520 1253 1549">Info: "Setting the Image Display Mode" on page 213</p> |
| IntegralHeight | <p data-bbox="581 1570 1406 1638">Purpose: 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 data-bbox="581 1663 878 1692">Category: Appearance</p> <p data-bbox="581 1717 980 1747">Used by: Combo Box, List Box</p> <p data-bbox="581 1772 1114 1801">Can be read by script at run time: No</p> <p data-bbox="581 1827 1344 1881">Info: "Avoiding Clipping of Items in the Simple Combo Box Control" on page 245</p> |

| Property | Purpose, category, usage and further information |
|-------------------|---|
| Layout* | <p>Purpose: Defines the way the radio buttons are arranged in the group (Horizontal or Vertical).</p> <p>Category: Appearance</p> <p>Used by: Radio Button Group</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Layout of the Radio Button Group Options" on page 241</p> |
| Locked | <p>Purpose: 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>Category: Appearance</p> <p>Used by: 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>Can be read by script at run time: Yes</p> <p>Info: "Locking and Unlocking Elements" on page 148</p> |
| MaxDropDownItems* | <p>Purpose: Defines the maximum number of items the drop-down list shows.</p> <p>Category: Appearance</p> <p>Used by: Combo Box</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Maximum Number of Items to Appear in the Combo Box Drop-Down List" on page 246</p> |
| Multiline* | <p>Purpose: Determines if the control shows several lines of text that automatically wrap up when reaching the right border of the control.</p> <p>Category: Appearance</p> <p>Used by: Edit Box</p> <p>Can be read by script at run time: No</p> <p>Info: "Configuring the Text to Wrap in an Edit Box Control" on page 243</p> |

| Property | Purpose, category, usage and further information |
|-----------------|--|
| Radius* | <p>Purpose: Defines the radius of the corners of the Rounded Rectangle.</p> <p>Category: Appearance</p> <p>Used by: Rounded Rectangle</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Setting the Radius of Rounded Rectangles" on page 210</p> |
| ReadOnly* | <p>Purpose: Determines if the user can type data into the edit box.</p> <p>Category: Appearance</p> <p>Used by: Edit Box</p> <p>Can be read by script at run time: No</p> <p>Info: "Configuring the Text to be Read-Only in an Edit Box Control" on page 244</p> |
| RelativeAnchor* | <p>Purpose: Relative anchor point of the source symbol. By default, this is 0,0.</p> <p>Category: Appearance</p> <p>Used by: Canvas</p> <p>Can be read by script at run time: No</p> <p>Info: "Size Propagation and Anchor Points" on page 43</p> |
| RelativeOrigin | <p>Purpose: Defines the relative origin as X, Y location. The location is relative to the center point of the element (0, 0).</p> <p>Category: Appearance</p> <p>Used by: 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>Can be read by script at run time: No</p> <p>Info: "Changing Points of Origin in the Properties Editor" on page 134</p> |

| Property | Purpose, category, usage and further information |
|------------|---|
| ShowToday* | <p>Purpose: Determines if today's date is shown on the calendar control.</p> <p>Category: Appearance</p> <p>Used by: Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: "Showing or Hiding Today's Date on a Calendar Control" on page 248</p> |
| Smoothing* | <p>Purpose: When False the graphics are rendered normally, when True graphics are rendered with anti-aliasing which produces a smoother appearing graphic.</p> <p>Category: Appearance</p> <p>Used by: Canvas</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Radius of Rounded Rectangles" on page 210</p> |
| Start | <p>Purpose: Defines the start of a line or H/V line as X, Y location.</p> <p>Category: Appearance</p> <p>Used by: Line, H/V Line</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting Start or End Points of a Line" on page 169</p> |
| StartAngle | <p>Purpose: 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>Category: Appearance</p> <p>Used by: 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, 2 Point Arc, 3 Point Arc</p> <p>Can be read by script at run time: No</p> <p>Info: "Changing Angles of Arcs, Pies and Chords" on page 219</p> |

| Property | Purpose, category, usage and further information |
|--------------|---|
| SweepAngle | <p>Purpose: Defines the ending angle of the Arc, Pie or Chord. This angle is always measured from the start angle.</p> <p>Category: Appearance</p> <p>Used by: 2 Point Pie, 3 Point Pie, 2 Point Chord, 3 Point Chord, 2 Point Arc, 3 Point Arc</p> <p>Can be read by script at run time: No</p> <p>Info: "Changing Angles of Arcs, Pies and Chords" on page 219</p> |
| Tension | <p>Purpose: Specifies how tightly the curve bends through the control points of the curve.</p> <p>Category: Appearance</p> <p>Used by: Closed Curve, Curve</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Changing the Tension of Curves and Closed Curves" on page 219</p> |
| Text | <p>Purpose: Defines the unicode text that is shown by the element.</p> <p>Category: Appearance</p> <p>Used by: Button, Text, Text Box, Edit Box</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Setting the Displayed Text" on page 171</p> |
| TextFormat | <p>Purpose: Defines the formatting string that is applied to the text when it is shown.</p> <p>Category: Appearance</p> <p>Used by: Button, Text, Text Box</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Setting the Text Display Format" on page 171</p> |
| Transparency | <p>Purpose: Defines the transparency. A value of 0 means fully opaque and a value of 100 means fully transparent.</p> <p>Category: Appearance</p> <p>Used by: 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>Can be read by script at run time: Yes</p> <p>Info: "Setting the Transparency Level of an Element" on page 186</p> |

| Property | Purpose, category, usage and further information |
|-------------------|---|
| TransparentColor* | <p>Purpose: Defines the RGB color value that is used as the transparent color.</p> <p>Category: Appearance</p> <p>Used by: Image</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Image Color Transparency" on page 214</p> |
| UpImage* | <p>Purpose: Defines the image that is used in the button element when it is un-clicked or released.</p> <p>Category: Appearance</p> <p>Used by: Button</p> <p>Can be read by script at run time: No</p> <p>Info: "Configuring Buttons with Images" on page 217</p> |
| Width | <p>Purpose: Defines the width of the element.</p> <p>Category: Appearance</p> <p>Used by: 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>Can be read by script at run time: Yes</p> <p>Info: "Resizing Elements by Changing Size Properties" on page 129</p> |
| WordWrap | <p>Purpose: 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>Category: Appearance</p> <p>Used by: Button, Text Box</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Wrapping Text in Buttons" on page 217</p> |

| Property | Purpose, category, usage and further information |
|----------|---|
| X | <p>Purpose: Defines the left position of the element.</p> <p>Category: Appearance</p> <p>Used by: 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>Can be read by script at run time: Yes</p> <p>Info: "Moving Elements" on page 121</p> |
| Y | <p>Purpose: Defines the top position of the element.</p> <p>Category: Appearance</p> <p>Used by: 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>Can be read by script at run time: Yes</p> <p>Info: "Moving Elements" on page 121</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 symbols.

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 symbol.

| Property | Purpose, category, usage and further information |
|---------------------|---|
| FillBehavior | <p>Purpose: Determines how the Fill (Horizontal, Vertical or Both) should be applied to the element.</p> <p>Category: Fill Style</p> <p>Used by: 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>Can be read by script at run time: No</p> <p>Info: "Setting Fill Behavior" on page 167</p> |
| FillColor | <p>Purpose: Defines the fill style used for the filled portion of the element.</p> <p>Category: Fill Style</p> <p>Used by: 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>Can be read by script at run time: No</p> <p>Info: "Setting Fill Style" on page 165 and "Changing Background Color and Text Color of Windows Common Controls" on page 239</p> |
| FillOrientation | <p>Purpose: Determines the orientation of the fill when the element orientation is any value other than 0.</p> <p>Category: Fill Style</p> <p>Used by: 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>Can be read by script at run time: No</p> <p>Info: "Setting Fill Orientation" on page 166</p> |
| HorizontalDirection | <p>Purpose: Determines the horizontal direction of the fill for the element. Can be "Right" or "Left".</p> <p>Category: Fill Style</p> <p>Used by: 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>Can be read by script at run time: No</p> <p>Info: "Setting Horizontal Fill Direction and Percentage" on page 167</p> |

| Property | Purpose, category, usage and further information |
|-----------------------|---|
| HorizontalPercentFill | <p>Purpose: Determines the percentage of horizontal fill for the element.</p> <p>Category: Fill Style</p> <p>Used by: 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>Can be read by script at run time: Yes</p> <p>Info: "Setting Horizontal Fill Direction and Percentage" on page 167</p> |
| TitleFillColor* | <p>Purpose: Determines the background solid color in the title bar of the calendar control.</p> <p>Category: Fill Style</p> <p>Used by: Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting Title Fill Color and Text Color on a Calendar Control" on page 248</p> |
| UnFilledColor | <p>Purpose: Determines the element's unfilled area appearance.</p> <p>Category: Fill Style</p> <p>Used by: 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>Can be read by script at run time: No</p> <p>Info: "Setting Unfilled Style" on page 166</p> |

| Property | Purpose, category, usage and further information |
|---------------------|--|
| VerticalDirection | <p>Purpose: Defines the vertical direction of the fill. Can be "Top" or "Bottom".</p> <p>Category: Fill Style</p> <p>Used by: 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>Can be read by script at run time: No</p> <p>Info: "Setting Vertical Fill Direction and Percentage" on page 168</p> |
| VerticalPercentFill | <p>Purpose: Determines the percentage of vertical fill for the element.</p> <p>Category: Fill Style</p> <p>Used by: 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>Can be read by script at run time: Yes</p> <p>Info: "Setting Vertical Fill Direction and Percentage" on page 168</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 symbols.

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 symbol.

| Property | Purpose, category, usage and further information |
|-------------|--|
| EndCap | <p>Purpose: Defines the cap used at the end of the line of an open element.</p> <p>Category: Line Style</p> <p>Used by: Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting Line End Shape and Size" on page 211</p> |
| LineColor | <p>Purpose: Defines the color and affects of the line or border.</p> <p>Category: Line Style</p> <p>Used by: 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>Can be read by script at run time: No</p> <p>Info: "Setting the Line Style" on page 170</p> |
| LinePattern | <p>Purpose: Defines the pattern of the line or border.</p> <p>Category: Line Style</p> <p>Used by: 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>Can be read by script at run time: No</p> <p>Info: "Setting the Line Pattern" on page 169</p> |

| Property | Purpose, category, usage and further information |
|------------|--|
| LineWeight | <p>Purpose: Determines the weight of the element's line or border. A value of 0 means that there is no line or border.</p> <p>Category: Line Style</p> <p>Used by: 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>Can be read by script at run time: Yes</p> <p>Info: "Setting the Line Weight" on page 169</p> |
| StartCap | <p>Purpose: Defines the cap used at the start of the line of an open graphic.</p> <p>Category: Line Style</p> <p>Used by: Line, H/V Line, Polyline, Curve, 2 Point Arc, 3 Point Arc</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting Line End Shape and Size" on page 211</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 symbols.

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 symbol.

| Property | Purpose, category, usage and further information |
|-----------|---|
| Alignment | <p>Purpose: Controls the location of the text relative to the bounding rectangle of the element.</p> <p>Category: Text Style</p> <p>Used by: Button, Text, Text Box</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Text Alignment" on page 173</p> |
| Caption* | <p>Purpose: 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>Category: Text Style</p> <p>Used by: Check Box</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Caption Text of a Check Box Control" on page 242</p> |
| Font | <p>Purpose: Defines the basic text font as defined by the operating system.</p> <p>Category: Text Style</p> <p>Used by: Button, Text, Text Box, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar, DateTime Picker, List Box</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Text Font" on page 172</p> |
| TextColor | <p>Purpose: Defines the color and affects applied to the text.</p> <p>Category: Text Style</p> <p>Used by: Button, Text, Text Box, Radio Button Group, Check Box, Edit Box, Combo Box, Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Text Color" on page 172 and "Changing Background Color and Text Color of Windows Common Controls" on page 239</p> |

| Property | Purpose, category, usage and further information |
|--------------------|--|
| TitleTextColor* | <p>Purpose: Determines the text solid color in the title bar of the calendar control.</p> <p>Category: Text Style</p> <p>Used by: Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting Title Fill Color and Text Color on a Calendar Control" on page 248</p> |
| TrailingTextColor* | <p>Purpose: Determines the text solid color of the text for the trailing days. The trailing days are days outside the current month.</p> <p>Category: Text Style</p> <p>Used by: Calendar</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Text Color for Trailing Dates in a Calendar Control" on page 249</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 symbols.

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 symbol.

| Property | Purpose, category, usage and further information |
|------------|--|
| Enabled | <p>Purpose: 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>Category: Runtime Behavior</p> <p>Used by: 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>Can be read by script at run time: Yes</p> <p>Info: "Enabling and Disabling Elements for Run-Time Interaction" on page 188</p> |
| Language | <p>Purpose: Defines the current language of the symbol.</p> <p>Category: Runtime Behavior</p> <p>Used by: Embedded Symbol</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Selecting the Language for a Symbol" on page 428.</p> |
| LanguageID | <p>Purpose: Defines the current language ID of the symbol.</p> <p>Category: Runtime Behavior</p> <p>Used by: Embedded Symbol</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Selecting the Language for a Symbol" on page 428.</p> |

| Property | Purpose, category, usage and further information |
|------------------------|--|
| MultiplePopupsAllowed* | <p>Purpose: 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>Category: Runtime Behavior</p> <p>Used by: Canvas</p> <p>Can be read by script at run time: No</p> <p>Info: "Setting the Radius of Rounded Rectangles" on page 210</p> |
| OwningObject* | <p>Purpose: Used as the ArcestrA 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>Category: Runtime Behavior</p> <p>Used by: Embedded Symbol</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Detecting and Editing the Containing AutomationObject Instance" on page 415</p> |
| Scripts* | <p>Purpose: Defines a collection of scripts configured for the symbol.</p> <p>Category: Runtime Behavior</p> <p>Used by: Canvas</p> <p>Can be read by script at run time: No</p> <p>Info: "Adding and Maintaining Symbol Scripts" on page 369</p> |

| Property | Purpose, category, usage and further information |
|------------------|---|
| SymbolReference* | <p>Purpose: 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> <hr/> <p>Note: This property is always disabled.</p> <hr/> <p>Category: Runtime Behavior</p> <p>Used by: Embedded Symbol</p> <p>Can be read by script at run time: No</p> <p>Info: "Detecting the Source Symbol of an Embedded Symbol" on page 411</p> |
| TabOrder | <p>Purpose: 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>Category: Runtime Behavior</p> <p>Used by: 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>Can be read by script at run time: No</p> <p>Info: "Editing the Tab Order of an Element" on page 189</p> |
| TabStop | <p>Purpose: Determines if the element can be navigated to and can receive focus at run time.</p> <p>Category: Runtime Behavior</p> <p>Used by: 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>Can be read by script at run time: No</p> <p>Info: "Editing the Tab Order of an Element" on page 189</p> |

| Property | Purpose, category, usage and further information |
|-------------|--|
| TreatAsIcon | <p>Purpose: If this property is set to False, the animations defined on the graphics within the group or embedded symbol take precedence over an animation defined on the group or embedded symbol. If there are no animations or the user clicked on an area of the group or embedded symbol that does not have an animation, then the group or embedded symbol animation executes.</p> <p>If the property is set to True, only the animation on the group or embedded symbol is executed. The interactive animations within the group or embedded symbol never execute.</p> <p>Category: Runtime Behavior</p> <p>Used by: Group, Embedded Symbol</p> <p>Can be read by script at run time: Yes</p> <p>Info: "Editing the Embedded Symbol" on page 408</p> |
| Visible | <p>Purpose: 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>Category: Runtime Behavior</p> <p>Used by: 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>Can be read by script at run time: Yes</p> <p>Info: "Changing the Visibility of Elements" on page 189</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 symbols.

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.

| Property | Purpose, category, usage and further information |
|------------------|---|
| CustomProperties | <p>Purpose: The collection of CustomProperties defined by the symbol.</p> <p>Category: Custom Properties</p> <p>Used by: Canvas, Embedded Symbol</p> <p>Can be read by script at run time: No</p> <p>Info: "Using Custom Properties" on page 253</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

You can use the methods of the Windows common controls to manipulate the controls at run time by using them in scripting.

Overview of Windows Common Control List Methods

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>Purpose: 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>Used by: Combo Box, List Box</p> <hr/> <p>Note: This function does not work when using an Enum or Array to populate the List Box.</p> <hr/> <p>Syntax: <i>ControlName.AddItem(CaptionString);</i></p> <p>Info: "Adding and Inserting Items into a List" on page 383</p> |
| Clear | <p>Purpose: Removes all items from the List. If the list is bound, it clears the bound reference (array or enum) in Arcestra.</p> <hr/> <p>Note: This function does not work when using an Enum or Array to populate the List Box.</p> <hr/> <p>Used by: Combo Box, List Box</p> <p>Syntax: <i>ControlName.Clear();</i></p> <p>Info: "Deleting Items from a List" on page 384</p> |
| DeleteItem | <p>Purpose: 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>Used by: Combo Box, List Box</p> <p>Syntax: <i>ControlName.DeleteItem(Index);</i></p> <p>Info: "Deleting Items from a List" on page 384</p> |
| DeleteSelection | <p>Purpose: Delete the currently selected item from the list.</p> <p>Used by: Combo Box, List Box</p> <p>Syntax: <i>ControlName.DeleteSelection();</i></p> <p>Info: "Deleting Items from a List" on page 384</p> |
| FindItem | <p>Purpose: 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>Used by: Combo Box, List Box</p> <p>Syntax: <i>ControlName.FindItem(SearchString);</i></p> <p>Info: "Finding an Item in a List" on page 385</p> |

| Method | Purpose, syntax and information |
|---------------|---|
| GetItem | <p>Purpose: 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>Used by: Combo Box, List Box</p> <p>Syntax: <i>ItemCaption = ControlName.GetItem(Index);</i></p> <p>Info: "Reading the Caption of a Selected Item in a List" on page 385</p> |
| InsertItem | <p>Purpose: Inserts the supplied string after the current selection in the List. Does not work if list is sorted.</p> <p>Used by: Combo Box, List Box</p> <p>Syntax: <i>ControlName.InsertItem(String);</i></p> <p>Info: "Adding and Inserting Items into a List" on page 383</p> |
| SetItemData | <p>Purpose: 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>Note: 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>Used by: Combo Box, List Box</p> <p>Syntax: <i>ControlName.SetItemData(Index,Value);</i></p> <p>Info: "Associating Items with Values in a List" on page 385</p> |
| GetItemData | <p>Purpose: 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>Note: 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>Used by: Combo Box, List Box</p> <p>Syntax: <i>Value = ControlName.GetItemData(Index);</i></p> <p>Info: "Associating Items with Values in a List" on page 385</p> |

| Method | Purpose, syntax and information |
|---------------|--|
| LoadList | <p>Purpose: 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>Note: The LoadList method does not work when using an Enum or Array to populate the List Box control.</p> <hr/> <p>Used by: Combo Box, List Box</p> <p>Syntax: <i>ControlName.LoadList(FileName);</i></p> <p>Info: "Loading and Saving Item Lists" on page 386</p> |
| LoadText | <p>Purpose: 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>Used by: Edit Box</p> <p>Syntax: <i>ControlName.LoadText(FileName);</i></p> <p>Info: "Configuring Edit Box Methods" on page 382</p> |
| SaveList | <p>Purpose: 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>Used by: Combo Box, List Box</p> <p>Syntax: <i>ControlName.SaveList(FileName);</i></p> <p>Info: "Loading and Saving Item Lists" on page 386</p> |
| SaveText | <p>Purpose: 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>Used by: Edit Box</p> <p>Syntax: <i>ControlName.SaveText(FileName);</i></p> <p>Info: "Configuring Edit Box Methods" on page 382</p> |

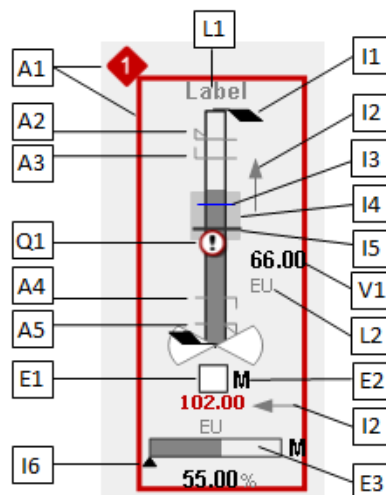
Appendix C






Situational Awareness Library Symbols



















The Archestra Symbol Editor includes a set of Situational Awareness Library symbols. These symbols include multiple visual and functional configurations that are enabled by selecting values from a set of Wizard Options associated with each symbol.








Common Graphic Elements of Situational Awareness Library Symbols




The following image shows a Situational Awareness Library symbol in which all graphic elements are shown. Typically, each configuration of a symbol only shows a subset of the graphic elements.




| Element | Description |
|---|---|
| A1  | Alarm Border Displays the state of the most urgent alarm configured for a symbol. The optional border and severity icons are inactive by default. Wizard Options: <ul style="list-style-type: none"> • SymbolMode Advanced • AlarmBorder True |
| A2  | Hi Hi Alarm Limit Indicator This optional indicator appears on the fill bar so operators can compare the current measured value in relationship to the limit. Wizard Options: <ul style="list-style-type: none"> • SymbolMode Advanced • AlarmLimitIndicators True • AlarmHiHiIndicator True |
| A3  | Hi Alarm Limit Indicator This optional indicator appears on the fill bar so operators can compare the current measured value in relationship to the limit. Wizard Options: <ul style="list-style-type: none"> • SymbolMode Advanced • AlarmLimitIndicators True • AlarmHiIndicator True |
| A4  | Lo Alarm Limit Indicator This optional indicator appears on the fill bar so operators can compare the current measured value in relationship to the limit. Wizard Options: <ul style="list-style-type: none"> • SymbolMode Advanced • AlarmLimitIndicators True • AlarmLoIndicator True |
| A5  | Lo Lo Alarm Limit Indicator This optional indicator appears on the fill bar so operators can compare the current measured value in relationship to the limit. Wizard Options: <ul style="list-style-type: none"> • SymbolMode Advanced • AlarmLimitIndicators True • AlarmLoLoIndicator True |

| Element | Description | | | | | | | | | | | | | | | | | | |
|---|---|---|---------|-------|---|--------|-------|---|--------------------------------|----------|---|--------------------------------|----------|---|-------------|-----|---|----------------|-----|
| E1 | <p>Status Indicators These optional indicators show the operating status of equipment. The setting of Status Level option determines the number of available indicators.</p> <table border="0"> <tr> <td data-bbox="732 443 769 485"></td> <td data-bbox="875 428 976 453">Passive</td> <td data-bbox="1175 428 1247 453">Basic</td> </tr> <tr> <td data-bbox="732 533 769 575"></td> <td data-bbox="875 518 959 543">Active</td> <td data-bbox="1175 518 1247 543">Basic</td> </tr> <tr> <td data-bbox="732 623 769 665"></td> <td data-bbox="875 609 1117 669">Active traveling to Passive</td> <td data-bbox="1175 609 1300 634">Advanced</td> </tr> <tr> <td data-bbox="732 714 769 756"></td> <td data-bbox="875 699 1133 760">Passive traveling to Active</td> <td data-bbox="1175 699 1300 724">Advanced</td> </tr> <tr> <td data-bbox="732 804 769 846"></td> <td data-bbox="875 789 1024 814">Interlocked</td> <td data-bbox="1175 789 1214 814">All</td> </tr> <tr> <td data-bbox="732 894 769 936"></td> <td data-bbox="875 879 1057 905">Out of Service</td> <td data-bbox="1175 879 1214 905">All</td> </tr> </table> <p>Wizard Options</p> <ul style="list-style-type: none"> • Symbol Mode Advanced • EquipmentStatusIndicator True • EquipmentStatusLevel Basic |  | Passive | Basic |  | Active | Basic |  | Active traveling to Passive | Advanced |  | Passive traveling to Active | Advanced |  | Interlocked | All |  | Out of Service | All |
|  | Passive | Basic | | | | | | | | | | | | | | | | | |
|  | Active | Basic | | | | | | | | | | | | | | | | | |
|  | Active traveling to Passive | Advanced | | | | | | | | | | | | | | | | | |
|  | Passive traveling to Active | Advanced | | | | | | | | | | | | | | | | | |
|  | Interlocked | All | | | | | | | | | | | | | | | | | |
|  | Out of Service | All | | | | | | | | | | | | | | | | | |
| E2 | <p>Mode Indicators Shows the operating mode of the equipment or controller. This option is only available when the associated equipment or controller is enabled.</p> <ul style="list-style-type: none"> • M Manual • A Automatic • C Cascade <p>Wizard Options:</p> <ul style="list-style-type: none"> • SymbolMode Advanced • ControlModeIndicator True • MotorType Variable • SpeedController True • SpeedControllerMode True | | | | | | | | | | | | | | | | | | |

| Element | Description |
|---|---|
| E3 At Minimum  In Between  At Maximum  | Controller Output Fillbar This optional element displays the controller output value in relationship to minimum and maximum scale values. Wizard Options: <ul style="list-style-type: none"> • MotorType Variable • SymbolMode Advanced • SpeedController True |
| I1  | Full Range Indicators These indicators appear when operating range is configured and the current measured value is not within the set operating range. The fill bar will be forced to full range so that the measured value is always visible. Wizard Options: <ul style="list-style-type: none"> • SymbolMode Advanced • FullRangeIndicator True |
| I2 PV ROC  Amps ROC  | Rate of Change Indicators These optional indicators appear when a change of the associated value exceeds the ROC percentage. They indicate that a change has occurred and in which direction the value changed. Wizard Options (PV ROC): <ul style="list-style-type: none"> • SymbolMode Advanced • ROCIndicator True Wizard Options (Amps ROC): <ul style="list-style-type: none"> • MotorType Variable • SymbolMode Advanced • AmpsPV True • AmpsROCIndicator True |
| I3  | Tracker Indicator This optional indicator displays a value set by the operator. Operators can use this setting to track the behavior of the measured value in relationship to the tracked value. Wizard Options: <ul style="list-style-type: none"> • SymbolMode Advanced • Tracker True |

| Element | Description |
|---|--|
| I4  | <p>Optimal Range This optional gray area highlights the value range in which a measured value is expected to operate to achieve optimal performance.</p> <p>Wizard Options:</p> <ul style="list-style-type: none"> • SymbolMode Advanced • OptimalRange True |
| I5  | <p>Set Point Indicator This optional indicator is shown on the fill bar to compare the current measured value to the desired set point value.</p> <p>Wizard Options:</p> <ul style="list-style-type: none"> • SymbolMode Advanced • SetPoint True |
| I6  | <p>Commanded Value Indicator This optional indicator appears on a fill bar and shows the relationship between a measured value and its commanded value. Not all graphics support a commanded value. The commanded value may be associated with different measured variables such as PV, ControllerOP, and SpeedControllerPV. This option is only available when the measured value is enabled.</p> <p>Wizard Options:</p> <ul style="list-style-type: none"> • MotorType Variable • SymbolMode Advanced • SpeedController True • SpeedControllerCMDValue True |
| L1 Label | <p>Descriptive Label This option text label shows the name of the equipment represented by a symbol. The label can be a static text string or associated with a dynamic reference value or expression.</p> <p>Wizard Options:</p> <ul style="list-style-type: none"> • Label True |

| Element | Description |
|---|--|
| L2 EU | <p>Engineering Unit Labels These optional labels display the engineering units of a value. The labels can be static text strings or associated with a dynamic reference values or expressions. The engineering unit's options become available when the property's numerical display setting is true.</p> <p>Wizard Options (PV):</p> <ul style="list-style-type: none"> • <u>PV</u>NumericalDisplay True • EngUnits True <p>Wizard Options (Amps or SpeedCtl):</p> <ul style="list-style-type: none"> • MotorType Variable • SymbolMode Advanced • <u>Amps</u>PV True • <u>Amps</u>EngUnits True |
| Q1  | <p>Quality Status Indicator This optional indicator shows any abnormal quality state. There are several icons that represent the different quality states.</p> <p>Wizard Options:</p> <ul style="list-style-type: none"> • QualityStatusIndicator True <p>Refer to the ArcestrA Galaxy Style Library configuration in the IDE for settings of quality states: Galaxy > Configure > Galaxy Style Library (Quality Style Tab)</p> |
| V1 66.00 | <p>Numerical Values These optional fields are the numeric display of the measured values. The numerical display options become available when the measured value type is enabled. The PV measured value is generally always enabled. Other measured values such as Amps or Speed Controller have to be enabled.</p> <p>Wizard Options (PV):</p> <ul style="list-style-type: none"> • PVNumericalDisplay True <p>Wizard Options (Amps, SpeedCtl):</p> <ul style="list-style-type: none"> • MotorType Variable • SymbolMode Advanced • AmpsPV True • SpeedControllerNumericalDisplay True |

Index

A

- .aaCFG
 - XML manifest resource list of dynamically loaded assemblies 401
- aaHistClientDatabase.dll 396
- .aaPKG file
 - exporting client controls in 389, 398
 - exporting symbols to 75
 - importing a previously exported 390
 - importing client controls from 389, 390
 - importing symbols and graphic toolsets 74
 - importing symbols from 74
- absolute point of origin 134
- AbsoluteAnchor property 412, 506, 531
- AbsoluteOrigin property 506, 531
- action scripts
 - importing with a SmartSymbol 422–423
 - starting with a key combination 65
 - triggers 330–331
 - using 66
 - using to add an item to a Combo Box control 383
 - using to configure an element with animation 331
 - using to load text into an Edit Box control 382
- ActiveFactory 388
- ActiveFactory TagPicker client control 390, 393
- AddItem() method 383, 556
- Alarm Border animation 293–298
- Alignment property 506, 548
- anchor points
 - changing position 412
 - hiding 413
 - setting 412
 - showing 413
- AnchorFixedTo property 506, 531
- AnchorPoint property 507, 531
- Angle property 133, 507, 532
- Animation Summary 270
- animations
 - adding to graphic element 270
 - Alarm Border 293–298
 - associating with client control container properties 397–398
 - comparing InTouch and ArcestrA 421–422
 - comparison between InTouch and ArcestrA Symbol Editor 63
 - configuration parameters by type 281
 - configuring action script 330–332
 - configuring analog value displays 312–313

- configuring analog value pushbutton 328–329
- configuring blink 292
- configuring Boolean fill style 204–205, 282–283
- configuring Boolean value displays 311
- configuring Boolean value pushbutton 327–328
- configuring Calendar Control 350
- configuring Check Box 345
- configuring Combo Box 347–349
- configuring DateTime Picker 351–352
- configuring disable 317
- configuring Edit Box 346
- configuring fill style 204–206, 282–285
- configuring height 307–308
- configuring hide symbol 340
- configuring horizontal location 305
- configuring horizontal slider 325
- configuring line style 285–288
- configuring List Box 352–355
- configuring name displays 316
- configuring orientation 309–310
- configuring percent fill horizontal 301–302
- configuring percent fill vertical 303–304
- configuring pushbutton 329–330
- configuring Radio Button Group 342–345
- configuring show symbol 332–339
- configuring status element 341–342
- configuring string value displays 314
- configuring string value pushbutton 329–330
- configuring text style 289–291
- configuring time value displays 314–315
- configuring tooltip 316–317
- configuring truth table fill style 205–206, 283–285
- configuring user input 318–324
- configuring value display 311–316
- configuring vertical location 306
- configuring vertical slider 326
- configuring visibility parameters 282
- configuring width 306–307
- connecting to data sources 279
- connecting to element properties 275–276
- connecting to element property references 276–277
- connecting with InTouchEventViewApp attributes 278
- cutting, copying, and pasting 366–367
- data type state 38
- element-specific 341
- enabling and disabling 272
- fill direction 301, 303
- fill orientation 301, 303
- hiding list 271
- importing from a SmartSymbol to ArchestrA graphic 421–422
- language switching 427
- managing 280
- point 308–309
- removing from a graphic element 271
- resetting default configuration values 273
- reviewing graphic element assignment 270
- showing list 271
- sorting the list 280
- states 38
- Sweep Angle 221
- switching between graphic elements 281
- truth table state 39
- types 269
- unfill color 301, 303
- validating configuration 272–273
- anti-aliasing filter 191
- application scripts 65
- ArchestrA package file
 - exporting client controls 398
 - exporting symbols to 75
 - importing previously exported client controls 390
 - importing symbols and graphic toolsets 74
- ArchestrA Symbol Editor
 - adding animations from Animation Summary 270
 - adding custom properties to a symbol 57
 - animation replication 56
 - animation summary 25
 - canvas 25
 - Canvas drawing area 59
 - combining elements into a group 57
 - comparison of animations to InTouchEvent 63
 - comparison of supported data types to InTouchEvent 62
 - configuring symbol editor preferences 103–104
 - converting data from InTouchEvent 62
 - creating graphics 58–60

- data sources 61
- description 19, 23
- differences between InTouch WindowMaker 55–58
- differences between WindowMaker 55–58
- disabling animations 272
- editing the source of an embedded symbol 411
- element positioning 57
- elements 25–30
- Elements List 24, 24–25
- embedding source symbols contained in an AutomationObject instance 407
- embedding source symbols contained in AutomationObject template 407
- enabling animations 272
- miscellaneous enhancements 58
- opening symbols in read-only mode 21
- panning and zooming the canvas 98–102
- Properties Editor 25
- redoing a specified number of undone changes 149
- removing an animation 271
- reviewing animations assigned to graphic elements 270
- setting color of drawing area 59
- setting style 176–186
- showing and hiding panels 98
- starting 97
- style replication 56
- supported InTouch mathematical functions 422
- supported InTouch string functions 422
- supported InTouch system functions 423
- tools and palettes 23
- Tools panel 24
- types of animations 63–64
- types of scripts 65
- undoing a single change 148, 149
- undoing a specified number of changes 149
- usability enhancements 56
- using animations 60–64
- using to configure predefined symbol scripts 376–377
- using to review which animations are assigned to elements 270
- viewing properties of embedded client controls 402
- Windows common controls 28–29

- arcs
 - changing sweep angle 220
 - setting starting point 220
 - setting sweep angle with handles 119
- arrowhead line end 211
- attributes
 - connecting animations 274
 - connecting animations with data sources 274
 - InTouchViewApp 278
 - reference mode 280
 - referencing a SuperTag 278
 - static mode reference 280
 - writing to attributes with Secured Write or Verified Write security 371
- auto image display mode 213
- AutomationObject instances
 - creating a copy containing an embedded symbol 416
 - creating symbols 68, 69–70
 - inheriting symbols from a template 21
 - location source of embedded symbol 415
 - selecting alternative embedded symbol 414
 - using for a single object 20
- AutomationObject templates
 - creating symbols 69
 - managing symbols 21
 - re-using symbols 68
 - using for multiple instances of symbols 20
- AutomationObjects
 - creating an instance by embedding an ArcestrA symbol 22
 - creating ArcestrA symbols in 21
 - hosting symbols 42
 - using me keyword to reference attributes 31
- AutoScale property 507, 532

B

- basic objects 59
- BindTo() method 265
- blink behavior 198, 199, 226, 227
- Boolean data type 61
- Boolean fill style animation 204–205, 282–283
- Boolean line style animation
 - configuring 286
 - setting line style and thickness 286
 - using default properties 286

- Boolean text style animation
 - configuring 289–290
 - setting default text style 290
 - using default style 290
- Brick pattern property 184
- buttons
 - automatically scaling text 216
 - configuring with images 217
 - description 216
 - down image 217
 - DownImage property 217
 - drawing 112
 - up image 217
 - using 216
 - wrapping text 217
- ButtonStyle property 507, 532
- buttonsUpImage property 217
- C**
- Calendar controls
 - changing colors 352
 - configuring 246–249
 - configuring animations 350
 - data type 240
 - description 28
 - description of date properties 246
 - resizing restrictions 128
 - setting date for first time view 249
 - setting first day of week 247
 - setting number of calendar month sheets 247
 - setting text color for trailing dates 249
 - setting title text color 248–249
 - showing or hiding current date 248
 - ShowToday property 521, 539
 - TitleFillColor property 524, 544
 - TitleTextColor property 524, 549
- CalendarColumns property 508, 532
- CalendarRows property 508, 533
- Canvas
 - description 59
 - dragging elements 114
- Caption property 508, 548
- Check Board pattern property 184
- Check Box animations 345
- Check Box control
 - configuring 242
 - ControlStyle property 243
- data type 240
- setting 3D appearance 243
- setting caption text 242
- setting default state 242
- Checked property 508, 533
- chords
 - changing start and sweep angle 220
 - setting sweep angle 220
- circle line end 211
- circular reference 40, 87
- Clear() method 384, 556
- client controls
 - ActiveFactory TagPicker 390
 - adding animation 398
 - animating 397–398
 - binding properties to attributes or element references 393–395
 - configuring event logging messages 397
 - configuring event scripts 395–397
 - description 30, 388
 - dynamically loaded assemblies, including during application deployment 399–402
 - embedding 391–392
 - embedding a symbol into an InTouch application 30
 - embedding into an ArcestrA symbol 391
 - exporting 398
 - exporting directly as ArcestrA package files 398
 - import failure message 389
 - importing 389–390
 - importing a previously exported ArcestrA package 390
 - importing previously exported client controls 390
 - installing ActiveFactory TagPicker 390
 - organizing 391
 - properties of element container 392
 - securing 398
 - translating 435
 - viewing additional information 402–403
 - viewing and changing properties 392–393
 - viewing assemblies 403
 - viewing class name, vendor, and version 403
 - viewing objects and referencing symbols 403
- color bar 177
- color disk 177

- Color Picker 178
- Color1 property 166, 509
- Color2 property 166, 509
- Color3 property 166, 509
- colors
 - adding to custom palette 179
 - analog alarm animation 64
 - analog fill animation 64
 - analog line animation 63
 - analog text animation 64
 - changing gradient variant 182
 - changing with element style 187
 - discrete alarm animation 63, 64
 - discrete fill animation 64
 - discrete line animation 63
 - discrete text animation 64
 - falloff distribution 183
 - gradient direction properties 181
 - gradient properties 180
 - hue property 177
 - luminance 177
 - removing from custom palette 179
 - RGB 178
 - saturation property 177
 - setting background of a Windows common control 239
 - setting center point of radial or point based gradient 184
 - setting custom gradient angle 182
 - setting distribution 183
 - setting fill gradient 166
 - setting focus scales 184
 - setting focus scales width 184
 - setting for text 172
 - setting from color disk and bar 177
 - setting from the custom palette 178
 - setting horizontal gradient 181
 - setting image transparency 214
 - setting image transparency color 214
 - setting number in gradient 180–181
 - setting pattern background 185
 - setting pattern foreground 185
 - setting radial gradient 182
 - setting solid from the Standard Palette 176–177
 - setting the drawing area 59
 - setting transparency 186
 - setting vertical gradient 181
 - setting with Color Picker 178
 - setting with value input boxes 177–178
 - standard palette 177
 - using for text animation 39
- Combo Box animations
 - configuring array and captions 348–349
 - configuring enum 349
 - configuring static 347–348
 - types 347
- Combo Box controls
 - clipping items 245
 - configuring 244–246
 - Count property 246
 - data types 240
 - description 28
 - loading an item list 386
 - MaxDropDownItems property 246
 - NewIndex 246
 - preventing clipping of items 245
 - saving an item list 386
 - setting maximum number of items in drop-down list 246
 - setting type 244–245
 - setting width of drop-down list 245
 - using properties at run time 246
- condition scripts 66
- Confetti pattern property 184
- Configuring
 - Show/Hide Graphic script function 448
- connection points
 - adding 138
 - changing position 138
 - description 135
- ConnectionType property 143
- connectors
 - adding connection points 138
 - changing length 144–145
 - changing shape 145
 - changing the position of a connection point 138
 - changing type 143
 - connection points 135
 - control point 135
 - description 135
 - drawing 136–137
 - importing and exporting with the programmatic API 136
 - properties 142–143

- Containment Relationships in scripts 463
 - control points
 - adding or removing 218–219
 - changing the tension of curves 219
 - description 135, 218
 - editing within a path graphic 157
 - moving 218
 - ControlStyle property 243, 509, 533
 - Count property 246, 252, 510
 - Cross pattern property 184
 - curves
 - adding control points 218
 - changing shape with control points 218
 - changing tension 219
 - deleting control points 218
 - drawing 111
 - moving control points 218
 - custom palette
 - adding colors to 179
 - description 178
 - loading 180
 - removing colors from 179
 - saving 179
 - custom properties
 - adding 254
 - adding to a symbols 57
 - analog historical summary data 260
 - changing the expression or reference at run time 267–268
 - configuring 255–257
 - connecting animations 276–277
 - deleting 254
 - description 30, 115, 253–254
 - Edit Custom Properties dialog box 254
 - external linking to InTouch tags 258
 - group properties 553–554
 - historical summary period 262
 - language switching 433, 443
 - linking to external sources 258
 - managing 254
 - moving to a group 410
 - overriding 259
 - overriding source symbol 409
 - public 40, 87
 - renaming 257
 - resetting configuration to default values 257
 - restoring default value and description 409
 - reverting to original 259
 - showing historical summary data 259–265
 - statistical historical summary data 261
 - translating 434
 - translations 434
 - using binding 265–266
 - using to extend functionality of a symbol 31
 - using to reference InTouch tag from ArchestrA symbol 277
 - validating 257
 - CustomFormat property 510, 533
 - CustomProperties property 510, 554
- ## D
- data change scripts 66
 - data formats in strings 363–366
 - Data Timeout function, OnShow script 377–378
 - DataStatus animations
 - configuring 341
 - restrictions 342
 - dates
 - setting customized format in DateTime Picker control 250
 - setting format in format string 364
 - setting long format of DateTime Picker control 250
 - DateTime Picker animations 351–352
 - DateTime Picker controls
 - AbsoluteOrigin() property 506
 - changing background color 239
 - configuring 249, 251
 - CustomFormat() property 510
 - data type 240
 - DefaultValue() property 510
 - description 29
 - Enabled() property 512
 - Font() property 513
 - resizing restrictions 128
 - setting date formats 250
 - setting default value 251
 - time formats 250
 - DefaultValue property 510, 533
 - DeleteItem() method 384, 556
 - DeleteSection() method 384
 - DeleteSelection() method 556
 - Description property 510, 530
 - Diagonals pattern property 184

- Diamon pattern property 184
- diamond line end 211
- discrete data type 61
- documentation conventions 17
- down image 217
- DownImage property 217, 511, 534
- DropDownType property 511, 534
- DropDownWidth property 511, 534
- dynamic size change 412, 413
- dynamically loaded assemblies
 - embedding XML manifest resource in primary assembly 401
 - including during application deployment 399–402
 - including XML manifest resource in a configuration file 401
 - preventing import issues 402
 - requirements 399
 - sample XML for list 400
 - XML schema for list 400
- DynamicSizeChange property 511, 534
- E**
- Edit Box animations
 - configuring 346
- Edit Box controls
 - configuring 243–244
 - data type 240
 - description 28
 - Editline property 244
 - LoadText() method 382, 558
 - SaveText() method 382, 558
 - setting default text 243
 - setting text to read-only 244
 - setting text word wrap 243
 - using methods to save and load text at run time 382
- elapsed time
 - data type 61
 - selecting as custom property 256
- element mode 153
- Element Style
 - applying from the Element Style list 201
 - applying from the Properties grid 202
 - applying to elements 201–203
 - applying to groups of elements 203–204
 - applying with format painter 202
 - changing user-defined 200
 - changing visual properties of element styles 196–199
 - element properties 194
 - fill visual properties 194
 - graphic elements 194
 - importing and exporting in the Galaxy Style Library 195
 - importing optional Galaxy Style Libraries 196
 - line visual properties 194
 - order of precedence 195
 - outline visual properties 194
 - previewing changes 199
 - resetting to default values 200
 - showing current styles of a Galaxy 196
 - text visual properties 194
 - updating at application runtime 195
 - using animation 195, 204–207
- Elements List 164
 - adding elements to group 151
 - description 24
 - selecting elements from 115, 117
 - using to change z-order of elements 131
- element-specific animations
 - description 269
 - supported elements 341
- ElementStyle property 143
- ellipses
 - basic element 25
 - closed element 26
 - drawing 110
 - editing line properties 168
- embedded symbols
 - appearance 39, 86
 - changing 40, 87
 - controlling size propagation 412
 - converting to a group 410
 - creating an new AutomationObject instance 416
 - description 21, 39, 86
 - editing 408–409
 - editing source 411
 - embedding from Graphic Toolbox 406–407
 - enabling dynamic size change 413–414
 - hiding anchor point 413
 - instantiating 41
 - language switching 431, 444

- locating AutomationObject instance that contains source symbol 415
- overriding custom properties 409
- propagating changes 42–44
- renaming source symbols 407–408
- restoring to original size of source symbol 410
- selecting alternate AutomationObject instances 415
- selecting alternate from AutomationObject instance 414
- showing anchor point 413
- translating custom properties 434
- viewing the source 411
- Enabled property 143, 512, 550
- End property 143, 512, 534
- EndCap property 143, 512, 546
- enumerations, setting format in format strings 366

F

- falloff color distribution 183
- falloff gradient
 - description 183
- FillBehavior property 167, 512, 543
- FillColor property 513, 543
- FillOrientation property 513, 543
- fills
 - blink behavior 198, 226
 - overriding appearance to indicate a change in quality 225
 - setting behavior 167
 - setting orientation 166
 - setting status override appearance for 223
 - setting style 165, 170
 - setting unfilled style 166
- FillStyle property 165
- FindItem() method 385, 556
- FirstDayOfWeek property 513, 535
- float
 - custom property 256
 - data type 61
 - setting as a text display 171
 - tension values 219
 - user input animation 318
- focus scales
 - description 184
 - setting width 184
- font

- language font at run time 442
- language switching 430
- Font property 172, 513, 548
- format painter
 - copying element formats to target elements 190
 - copying format of an element to another 191
 - copying the format of an element in repetitive mode 191
 - description 190
 - using with Element Styles 202
- Format property 514, 535
- format strings 363–366
 - dates 364
 - enumerations 366
 - examples 366
- functions
 - importing from
 - aaHistClientDatabase.dll 396
 - InTouch miscellaneous 423
 - InTouch string 423
 - InTouch system 423
 - InTouch types that can be imported to ArchestrA Symbol Editor 422–423

G

- Galaxy Browser 242, 246
 - browsing for List Box control 252
 - browsing properties in the List Box control 252
 - InTouch Tag Browser Tab 278
 - using to browse methods of the Combo Box control 347
 - using to browse methods of the Edit Box control 346
 - using to browse methods of the List Box control 353
 - using to connect animations to attribute references 274
 - using to connect animations to element properties 275
 - using to connect animations to element property references 275–276, 276–277
 - using to connect animations to InTouch tags 279
 - using to embed a client control into an ArchestrA symbol 391–392
 - using to embed symbols from the Graphic Toolbox 407
 - using to select an alternate instance 415

- using to select an alternate symbol 414
- using to show and select properties of elements or symbols 381
- Galaxy Style Library
 - Element Style 194
 - importing and exporting 196
- GetItem() method 385, 557
- GetItemData() method 386, 557
- gradients
 - changing variant 182
 - falloff 183
 - peak 183
 - properties 180
 - setting custom angle 182
 - setting direction 181
 - setting focus scales 184
 - setting horizontal direction 181
 - setting number of colors 180
 - setting transparency 186
 - setting vertical direction 181
 - triangular 183
- graphic elements
 - absolute point of origin 134
 - adding animations 270
 - adding control points 218
 - adding outlines to indicate non-good status or quality 198–199, 227
 - adding to an existing group 151
 - adding to path graphics 160
 - adjusting space between 125–128
 - adjusting the z-order 131
 - aligning 123–125
 - aligning by centers 124
 - aligning by points of origin 125
 - aligning horizontally 123
 - aligning vertically 124
 - appearance properties 530–542
 - basic 26
 - binding client control properties 394
 - bringing one level forward 131
 - bringing to front 131
 - changing angles of arcs, pies, and chords 219–220
 - changing properties with scripts 381
 - changing start and sweep angle of an arc, pie, or chord 220
 - changing tab order 189–190
 - changing the tension of curves 219
 - changing visibility 189
 - changing z-order within path graphic 159
 - closed 26
 - configuring 341
 - configuring analog value display
 - animations 312–313
 - configuring analog value pushbutton
 - animations 328–329
 - configuring analog value user input
 - animations 319–320
 - configuring animation fill style 204–206, 282–285
 - configuring animation line style 285–288
 - configuring animation text style 289–291
 - configuring blink animation 292
 - configuring Boolean value display
 - animations 311
 - configuring Boolean value pushbutton
 - animations 327–328
 - configuring disable animations 317
 - configuring discrete value user input
 - animations 318–319
 - configuring elapsed time value user input
 - animations 324
 - configuring height animations 307–308
 - configuring hide symbol animations 340
 - configuring horizontal location
 - animations 305
 - configuring horizontal slider
 - animations 325
 - configuring name display animations 316
 - configuring orientation animations 309–310
 - configuring percent fill horizontal
 - animations 301–302
 - configuring percent fill vertical
 - animations 303–304
 - configuring show symbol animations 317, 337–339
 - configuring string value display
 - animations 314
 - configuring string value pushbutton
 - animations 329–330
 - configuring string value user input
 - animations 321–322
 - configuring time value display
 - animations 314–315
 - configuring time value user input
 - animations 322–324
 - configuring tooltips 316–317
 - configuring vertical location animations 306

- configuring vertical slider animations 326
- configuring width animations 306–307
- connecting animations to ArchestrA attributes 274
- connecting animations to custom properties 276–277
- connecting animations to element properties 275–276
- connecting animations to InTouch tags 277–279
- control points 218–219
- copying and pasting animations 366
- copying locked and grouped 119
- copying, cutting, and pasting 119–121
- creating a group 150
- custom properties 553–554
- cutting and pasting 120
- cutting and pasting animations 367
- decreasing space between 127
- deleting 120
- deleting control points 218
- description 55, 109–110
- distributing 126
- dragging 114
- dragging and drawing 110–114
- duplicating 121
- editing controls points within path graphic 157
- editing fill properties 164–168
- editing line properties 168–170
- editing properties 114–115
- editing start or sweep angle within path graphic 156
- editing the name 164
- editing the tab order 189–190
- enabling and disabling animations 272
- enabling or disabling for run-time interaction 188–189
- fill style properties 542–545
- flipping 148
- flipping horizontally 148
- flipping vertically 148
- graphic properties 530
- handles 116
- hiding animation list 271
- hiding at run time 189
- increasing space between 127
- inline editing 118, 119
- InTouch types that can be imported with restrictions 420
- language switching 426
- line style properties 545–547
- locking 148
- locking and unlocking 148–149
- moving 121–122
- moving by specifying X and Y properties 122
- moving control points 218
- moving points of origin with mouse 134
- moving the origin 134–135
- moving with keyboard 122
- moving with mouse 122
- moving within path graphic 155–156
- open 26
- overriding appearance depending on quality and status attributes 196–200, 224–228
- overriding fill appearance to indicate non-good status or quality 198, 225–226
- positioning with ArchestrA Symbol Editor 57
- properties 505–529
- reference mode 280
- relative point of origin 134
- removing all space between 128
- removing animations 271
- removing from a group 152
- removing from path graphic 161
- resetting default animation values 273
- resizing 128–130
- resizing by changing size properties 129
- resizing proportionally 130
- resizing to same height 130
- resizing to same size 130
- resizing to same width 130
- resizing with mouse 129
- resizing within path graphics 156
- reviewing assigned animations 270
- rotating 132–133
- rotating 90 degrees 133
- rotating by changing Angle property 133
- rotating with mouse 132
- run-time behavior properties 549–553
- selecting 115–118
- selecting all with Select All function 117
- selecting by lasso 117
- selecting by mouse click 116

- selecting with Elements List 117
- sending one level backward 131
- sending to back 131
- setting colors and transparency of a gradient 187
- setting equal space between 126
- setting fill behavior 167
- setting fill gradient 166
- setting fill orientation 166
- setting line end shape and size 211
- setting overlap when duplicated 121
- setting text alignment 173
- setting text auto scaling 212
- setting text color 172
- setting text font 172
- setting text properties 171–175
- setting text word wrap 212
- setting the radius of rounded rectangles 210
- setting transparency level 186–187
- setting unfilled style 166
- showing animation list 271
- showing at run time 189
- static mode reference 280
- substituting references 367–368
- supported InTouch types that can be imported to ArcestrA 419–420
- swapping end points within path graphic 158
- switching between animations 281
- task list 109–110
- text style properties 547–549
- ungrouping 151
- unlocking 148
- unselecting 118
- using the format painter 190–191
- using undo and redo 149
- validating animation configuration 272–273
- working with groups 150–153
- Graphic Toolbox
 - creating copies of symbols 72
 - creating generic symbols 68
 - creating Graphic Toolsets 71
 - creating symbols 68
 - customizing Graphic Toolsets 74
 - deleting Graphic Toolsets 73
 - editing hosting symbols 42
 - embedding source symbols 407
 - managing symbols 20
 - moving Graphic Toolsets 74
 - moving symbols to different Graphic Toolsets 72
 - renaming Graphic Toolsets 73
 - Toolsets folder 20
 - using to store ArcestrA symbols 20
 - viewing class name, vendor, and version of a client control 403
 - viewing client control assemblies 403
 - viewing read-only symbols 96
 - viewing referencing AutomationObjects and symbols of client controls 403
- Graphic Toolsets
 - creating 71
 - customizing 74
 - deleting 73
 - moving 74
 - moving symbols 72
 - renaming 73
- GraphicAccess API 77–79
- GraphicInfo, Conflicting Parameters 452
- Graphics Performance Index 44
- Grid pattern property 184
- groups
 - adding elements to 151
 - converting embedded symbols 410
 - creating 150
 - deleting 120
 - description 60
 - editing components within 152–153
 - moving custom properties 410
 - properties of 32
 - removing elements from 152
 - selecting elements from 119
 - ungrouping 151
- H**
- H/V lines
 - adding to path graphic 160
 - drawing 110
 - editing 168
 - setting start and end points 169
- handles
 - description 116
 - primary 116
 - secondary 116
- HasTransparentColor property 212, 514, 535
- Height property 128, 514, 535

- hide symbol animations 340
 - Hierarchical References in scripts 463
 - historical summary data
 - analog statistics 260
 - description 259
 - showing 263–265
 - state statistics 261
 - summary period 262
 - horizontal alignment 123
 - Horizontal pattern property 184
 - horizontal scroll bar
 - configuring 252
 - displaying with HorizontalScrollBar property 515
 - HorizontalDirection property 514, 543
 - HorizontalPercentFill property 515, 543, 544
 - HorizontalScrollBar property 515, 536
 - HorizontalScrollbar property 252
- I
- IDE
- configuring security 87
 - creating a new symbol 68
 - deriving an instance from a template 21
 - description 17
 - symbol management tasks 67
 - symbol management tools 67
- Image property 216, 515, 536
- ImageAlignment property 214, 515, 536
- images
- adding editing application 215–216
 - alignment positions 214
 - auto sizing 213
 - configuring for buttons 217
 - description 212
 - editing 215
 - enabling color transparency 214
 - importing 112
 - placing on canvas 213
 - properties 212
 - selecting replacement 216
 - setting alignment 214
 - setting color transparency 214
 - setting display mode 213
 - setting editing application 215, 215–216
 - setting frame size 214
 - setting status override appearance for 223
 - setting transparency color 214
 - stretching 213
 - supported file formats 185, 213
 - tiled 213
- ImageStyle property 516, 536
- inline editing 118
- InsertItem() method 383, 557
- IntegralHeight property 245, 251, 516, 536
- interaction animations
 - description 269
 - types 281
- InternationalizedString data type 61
- InTouch
- animations 63–64
 - basic objects 59
 - binding run-time behavior with
 - animations 34
 - comparing script types to ArcestrA scripts 65–66
 - comparison of animations to ArcestrA Symbol Editor 63
 - comparison of supported data types to ArcestrA Symbol Editor 62
 - configuring data source 61
 - connecting element animations to tags 277–279
 - connecting tags to ArcestrA symbols 61
 - converting data to ArcestrA Symbol Editor 62
 - data types 61
 - differences between WindowMaker and ArcestrA Symbol Editor 55
 - editing a symbol embedded in a window 71
 - embedded symbol containing a client control 30
 - embedding symbol into a managed application 395
 - embedding symbols 21
 - functions supported by ArcestrA Symbol Editor 423
 - graphic objects not included in ArcestrA Symbol Editor 58
 - importing functions to ArcestrA Symbol Editor 422–423
 - importing SmartSymbol animation links 421–422
 - importing SmartSymbols into an ArcestrA symbol 60
 - importing SmartSymbols into ArcestrA symbols 417–419
 - instantiating a symbol 22

- keyword 37, 423
 - linking symbol custom properties to tags 258
 - mathematical functions supported by ArchestrA Symbol Editor 422
 - overview of ArchestrA integration 19
 - reference syntax to link a symbol to a tag 258
 - restrictions to importing SmartSymbols to ArchestrA Symbol Editor 419–420
 - starting ArchestrA Symbol Editor 97
 - string functions supported by ArchestrA Symbol Editor 422
 - supported graphic types that can be imported to ArchestrA 419–420
 - system functions supported by ArchestrA Symbol Editor 423
 - types of animations 63–64
 - types of scripts 65
 - updating embedded symbols in WindowMaker 42
 - using custom properties to reference a tag 31
 - using custom properties with tags 253
 - using tag values in a symbol 37
 - wizards 60
 - InTouch wizards 58
 - InTouchViewApp
 - connecting animations with attributes 278
 - connecting attributes to animations 278
 - managed InTouch application object 277
- K**
- key scripts 65
 - keywords
 - InTouch 37, 423
 - self-reference me 30
- L**
- language
 - removing for a symbol 429
 - selecting for a symbol 428
 - Language property 516, 550
 - language switching
 - ArchestrA symbols 425
 - custom properties 433, 443
 - data types 446
 - embedded symbols 431, 444
 - example 437
 - font 430
 - graphic elements 426
 - overriding translated custom properties 440
 - overriding translated string substitutions 440
 - overriding translations in WindowMaker 440
 - popup symbols 445
 - precedence rules 442
 - propagation of language setting changes 445
 - run time 440
 - SmartSymbols 436
 - string substitution 432
 - LanguageID property 516, 550
 - lasso 115, 117
 - Layout property 516, 537
 - LineColor property 143, 517, 546
 - LinePattern property 143, 517, 546
 - lines
 - ArchestrA Symbol Editor basic element 23
 - blink behavior 199, 227
 - element of a symbol 25
 - end shapes 211
 - overriding appearance to indicate a change in quality 221
 - overriding appearance to indicate non-good status or quality 198, 226
 - selecting tool from Tools panel 24
 - setting end shape and size 211
 - setting size of arrowheads 211
 - setting status override appearance for 222–223
 - used in path graphics 29
 - LineWeight property 143, 517, 547
 - List Box animations
 - configuring array and captions 354
 - configuring enum 354–355
 - configuring static 353
 - types 352
 - List Box controls
 - adding an item 383
 - associating items with values in a list 385
 - avoiding clipping of items 251
 - configuring 251–252
 - Count property 252
 - data type 240
 - deleting a selected item from a list 384
 - deleting all items from a list 384

- description 28
- finding an item in a list 385
- getting item data 386
- inserting an item in a list 383
- loading the item list 386
- NewIndex property 252
- reading the caption of a selected item in a list 385
- saving the item list 386
- SelectedValue property 252
- setting item data 386
- TopIndex property 252
- using horizontal scroll bar 252
- using properties at run time 252

LoadList() method 386, 558

LoadText() method 382, 558

Locked property 517, 537

M

managed application 278

MaxDropDownItems property 246, 518, 537

message data type 61

messages

- edit symbol confirmation 71
- Not Found after deleting embedded symbol 84
- security after user attempts to export a symbol without appropriate permissions 88
- warning when overwriting colors in a custom palette 180

methods

- AbsoluteOrigin property 506
- AddItem() 383, 556
- BindTo() 265
- Clear() 384, 556
- configuring Combo Box and List Box 383–386
- configuring Edit Box 382
- DeleteItem() 384, 556
- DeleteSection() 384
- DeleteSelection() 556
- FindItem() 385, 556
- GetItem() 385, 557
- GetItemData() 386, 557
- InsertItem() 383, 557
- LoadList() 386, 558
- LoadText() 382, 558
- SaveList() 386, 558

- SaveText() 382, 558
- SetItemData() 386, 557
- using in scripting 382

Modal Windows, working with 461

Multiline property 244, 518, 537

MultiplePopupsAllowed property 518, 551

N

Name property 518, 530

named scripts

- removing from a symbol 380
- renaming 380

names

- changing in Properties Editor 164
- changing in the Elements List 164

NewIndex property 246, 252, 519

normal image display mode 213

O

OnShow script Data Timeout function 377–378

outlines

- adding to elements to indicate non-good status or quality 198–199, 227
- description as quality status indicator 221
- replacing group handle 119
- resetting to default 228

OwningObject property 415, 519, 551

P

pan

- description 102
- using mouse 102

path graphics 29

- adding elements 160
- breaking the path 155
- changing 155–160
- changing element z-order 159–160
- creating 154
- description 55, 153
- editing element control points within 157
- editing start or sweep angle of element within 156
- moving elements within 155–156
- removing elements 161
- resizing element within 156
- swapping element end points within 158
- viewing 153

- path mode 153
- patterns
 - properties 184
 - setting 185
 - setting transparency 186
- peak gradient 183
- Percent pattern property 184
- pies
 - changing sweep angle 220
 - setting starting point 220
- Point animation 308–309
- point based gradient 184
- points of origin 125
- polylines
 - basic element of ArcestraA Symbol Editor 26
 - drawing 111
 - inline editing 118
 - moving control points 218
 - part of path graphics 29
 - shaping with control points 119
- popup symbols
 - language switching 445
- pop-ups 192
- properties
 - AbsoluteAnchor 412, 506, 531
 - AbsoluteOrigin 531
 - Alignment 506, 548
 - AnchorFixedTo 506, 531
 - AnchorPoint 507, 531
 - Angle 133, 507, 532
 - AutoScale 507, 532
 - binding client control attributes 393–394
 - ButtonStyle 507, 532
 - CalendarColumns 508, 532
 - CalendarRows 508, 533
 - Caption 508, 548
 - changing element tab order with 190
 - Checked 508, 533
 - client control container 392, 397
 - Color1 166, 509
 - Color2 166, 509
 - Color3 166, 509
 - configuring fill style with 165
 - ConnectionType 143
 - ControlStyle 243, 509, 533
 - Count 246, 252, 510
 - CustomFormat 510, 533
 - CustomProperties 510, 554
 - DefaultValue 510, 533
 - Description 510, 530
 - DownImage 217, 511, 534
 - DropDownType 511, 534
 - DropDownWidth 511, 534
 - DynamicSizeChange 511, 534
 - ElementStyle 143
 - Enabled 143, 512, 550
 - End 143, 512, 534
 - EndCap 143, 512, 546
 - FillBehavior 167, 512, 543
 - FillColor 513, 543
 - FillOrientation 513, 543
 - FillStyle 165
 - FirstDayOfWeek 513, 535
 - Font 172, 513, 548
 - Format 514, 535
 - HasTransparentColor 212, 514, 535
 - Height 128, 514, 535
 - HorizontalDirection 514, 543
 - HorizontalPercentFill 515, 543, 544
 - HorizontalScrollBar 515, 536
 - HorizontalScrollbar 252
 - Image 216, 515, 536
 - ImageAlignment 214, 515, 536
 - ImageStyle 516, 536
 - IntegralHeight 245, 251, 516, 536
 - Language 516, 550
 - LanguageID 516, 550
 - Layout 516, 537
 - LineColor 143, 517, 546
 - LinePattern 143, 517, 546
 - LineWeight 143, 517, 547
 - Locked 517, 537
 - MaxDropDownItem 518
 - MaxDropDownItems 246, 537
 - Multiline 244, 518, 537
 - MultiplePopupsAllowed 518, 551
 - Name 518, 530
 - NewIndex 246, 252, 519
 - OwningObject 415, 519, 551
 - Radius 519, 538
 - ReadOnly 244, 519, 538
 - RelativeAnchor 412, 520, 538
 - RelativeOrigin 520, 538
 - Scripts 520, 551
 - SelectedValue 252, 520

- selecting replacement image with 216
- setting fill behavior 167
- setting fill gradient with 166
- setting text display format with 171
- setting text font with 172
- ShowToday 521, 539
- Smoothing 521, 539
- Start 143, 521, 539
- StartAngle 521, 539
- StartCap 143, 522, 547
- SweepAngle 522, 540
- SymbolReference 411, 522, 552
- TabOrder 522, 552
- TabStop 190, 523, 552
- Tension 523, 540
- Text 523, 540
- TextColor 523, 548
- TextFormat 171, 524, 540
- TitleFillColor 524, 544
- TitleTextColor 524, 549
- topIndex 524
- TrailingTextColor 525, 549
- Transparency 525, 540
- TransparentColor 212, 215, 525, 541
- TreatAsIcon 526, 553
- UnFillColor 526, 544
- UpImage 217, 526, 541
- using Combo Box control at run time 246
- using List Box controls at run time 252
- Value 239, 526
- VerticalDirection 527, 545
- VerticalPercentFill 527, 545
- Visible 143, 527, 553
- Width 128, 528, 541
- WordWrap 217, 528, 541
- X 122, 529, 542
- Y 122, 529, 542
- Properties Editor
 - changing absolute point of origin 134
 - changing element names 164
 - changing relative point of origin 134
 - description 25
 - editing properties with 114
- pushbutton animations
 - analog value 328–329
 - Boolean value 327–328
 - string value 329–330

Q

- quality
 - adding outlines to elements to indicate status or non-good quality 199, 227
 - changing status to run named scripts 370
 - including Status element to show icon 25
 - monitoring symbol 221
 - overriding element appearance depending on attributes 196, 224
 - overriding element fill appearance to indicate non-good status 197–198, 225–226
 - overriding line appearance to indicate non-good quality or status 198, 226
 - overriding text appearance to indicate non-good status 197, 224–225
 - previewing all override appearances 199, 227–228
 - previewing override appearances 227–228
 - resetting override appearances to default 200, 228
 - showing 53
 - showing by overriding animations 53
 - showing for elements 24
 - showing symbol status 221
 - using DataStatus animation 341
 - using Status element to create icon 222
 - using Status element to show status 27–28
 - using status to add outlines to elements 199, 227
 - using status to override appearance of lines 198, 226
 - using status to override fill appearance of elements 197, 225
 - using status to override text appearance of elements 224

R

- radial gradients 182, 184
- Radio Button Group
 - configuring 240–242
 - data types 240
- Radio Button Group animations
 - configuring array and captions 343–344
 - configuring enum 344–345
 - configuring static 342–343
 - types 342
- Radio Button Group controls
 - configuring 240–241
 - setting 3D appearance 241

- setting layout of options 241
 - using properties at run time 241
 - Radius property 519, 538
 - ReadOnly property 519, 538
 - Readonly property 244
 - real
 - data type 61
 - imported local tag types 423
 - supported data type for Windows common controls 240
 - used in SuperTags 423
 - rectangles
 - basic object 24
 - closed object 26
 - drawing 110
 - elements of an ArcestrA symbol 23
 - fill style 186
 - selecting as basic object from ArcestrA Symbol Editor 59
 - references
 - animation 25, 34
 - AutomationObjects 75
 - circular 40, 87
 - element 30
 - keyword 30
 - substituting in elements 367–368
 - to InTouch tag 31
 - to InTouch tags 61
 - relative point of origin 134
 - RelativeAnchor property 412, 520, 538
 - RelativeOrigin property 520, 538
 - RGB color properties 178
 - rounded rectangles
 - description 210
 - enlarging radius 210
 - reducing radius 210
 - setting exact radius 210
- S**
- satellite assemblies
 - translations 435
 - SaveList() method 386, 558
 - SaveText() method 382, 558
 - scripts
 - action triggers 330–331
 - adding to a symbol 378–379
 - application 65
 - binding 265
 - changing properties of graphic elements 381
 - condition 66
 - configuring animations 330–331
 - configuring for client controls 395–397
 - configuring predefined of a symbol 376–377
 - configuring to symbols 369
 - data change 66
 - description 65–66, 369–370
 - editing for symbols 379
 - error handling 376
 - execution order 370
 - importing from a SmartSymbol 422–423
 - importing functions from
 - aaHistClientDatabase.dll 396
 - key 65
 - named 370
 - OnShow script execution 377–378
 - predefined 370
 - removing from an ArcestrA symbol 380
 - renaming a named script 380
 - security 371
 - setting time-out period for symbol
 - script 375
 - substituting attribute references 381
 - using 65–66
 - using methods 382
 - using with Combo box methods 383–386
 - using with Edit Box methods 382
 - using with List Box methods 383–386
 - While Showing 376
 - Scripts property 520, 551
 - Secured Write 371
 - security
 - configuring for symbols 87–88
 - enforcing by user role and permissions 88
 - IDE user permissions 87–88
 - in client controls 398
 - in scripts 371
 - warning message after attempting to export symbol without appropriate permissions 88
 - SelectedValue property 252, 520
 - SetCustomPropertyValue() method 267
 - SetItemData() method 386, 557
 - show symbol animations
 - configuring 337–339
 - element attributes 332–337
 - Show/Hide Graphic

- working with 447
- ShowToday property 521, 539
- Situational Awareness Library symbols
 - configuring in WindowMaker 491–493
 - description 85
 - functional configuration 85
 - graphic element description 559–564
 - protected symbols 475
- SmartSymbols
 - importing action scripts 422–423
 - importing into an ArcestrA symbol 60, 417–419
 - importing tags and references 423–424
 - importing tags and references to ArcestrA symbols 423–424
 - restrictions to importing 419–424
 - translating 436
- Smoothing property 521, 539
- square line end 211
- standard palette 177
- Start property 143, 521, 539
- StartAngle property 220, 521, 539
- StartCap property 143, 522, 547
- Status elements
 - associating with attributes and animated elements 222
 - configuring DataStatus animations 341
 - description 53, 55, 221
 - drawing 113
 - drawing icon on canvas 222
 - overriding element blink behavior 197, 225
 - overriding element fill appearance 197, 225
 - previewing all override appearances 199, 227–228
 - resetting to default appearance 228
 - restrictions of DataStatus animations 342
 - setting appearance 222
 - setting default appearance 222–223
 - using 221–224
 - using to indicate quality 27–28
- stretch image display mode 213
- string
 - comparison between InTouch and ArcestrA Symbol Editor animation 63
 - data type 61, 240
 - finding 175
 - format 363–366
 - language switching 432
 - matching 175
 - replacing 175
 - substituting 174
 - substituting in text properties of an element 171
- Sweep Angle animation 221
- SweepAngle property 220, 522, 540
- symbol scripts
 - actions against symbols 369
 - adding named type to a symbol 378–379
 - configuring predefined 376–377
 - editing 379
 - error handling 376
 - error messages from syntax errors 376
 - execution order 370
 - named 370
 - predefined 370
 - removing from a symbol 380
 - renaming named type 380
 - running when a condition becomes true 66
 - security 371
 - setting time-out period 375
 - writing to attributes with Secured Write or Verified Write security 371
- Symbol Wizard
 - Choice Groups 476
 - Choices 476
 - configuration rules 478–479
 - Consumer work flow 86
 - description 85, 475
 - Designer work flow 86
 - designing 480–488
 - embedding 490
 - exporting programmatically to an XML file 76
 - functional configuration 85
 - Graphic Performance Index 45
 - importing programmatically from an XML file 76
 - launching 480
 - layers 477
 - Options 476
 - verify configurations 489
 - visual configuration 85
 - work flows 86
- Symbol Wizard Editor
 - description 105–106
 - Layers view 476

- Options view 476
 - SymbolReference property 411, 522, 552
 - symbols
 - adding custom properties with ArcestrA Symbol Editor 57
 - adding named scripts 378–379
 - adding Windows common controls 238–239
 - allowing multiple pop-ups 192
 - animations 34–39
 - appearance of embedded 39, 86
 - changing embedded properties 40, 87
 - changing position of an anchor point 412
 - configuring predefined scripts 376–377
 - configuring security 87–88
 - controlling size propagation of embedded 412
 - creating in AutomationObject instances 69–70
 - creating in AutomationObject templates 69
 - creating in Graphic Toolbox 68
 - creating with the ArcestrA Symbol Editor 23
 - creation methods 68
 - data types 61–63
 - deleting 84
 - description 19, 67
 - editing contained in an AutomationObject 70–71
 - editing description 192
 - editing embedded into an InTouch window 71
 - editing general properties 191–192
 - editing scripts 379
 - embedded description 39, 86
 - embedding 406–407
 - embedding client controls 391
 - exporting programmatically 76–79
 - exporting to a .aaPKG file 75
 - hosted by AutomationObject 42
 - hosted by the Graphic Toolbox 42
 - importing from a .aaPKG file 74
 - importing programmatically 76–79
 - importing SmartSymbols 417–419
 - importing SmartSymbols tags and references 423–424
 - inherited 21
 - instantiating embedded 41
 - instantiating with an AutomationObject template 22
 - language settings for Galaxies 430
 - language switching 425
 - managed by Graphic Toolbox 20
 - managing 20
 - managing in AutomationObjects 21
 - monitoring and showing quality and status 221–224
 - monitoring quality and status 221
 - moving between Graphic Toolsets 72
 - opening for editing 70
 - organizing into Toolsets 20
 - placement in ArcestrA environment 20
 - propagating changes 42
 - properties 30–34
 - removing languages 429
 - removing scripts 380
 - renaming 72
 - renaming scripts 380
 - renaming source and hosting AutomationObjects 407–408
 - restrictions to importing SmartSymbols 419–424
 - re-using 21–22
 - script triggers 65
 - selecting a language 428
 - setting anchor point 412
 - setting the radius of rounded rectangles 210
 - showing quality and status 53
 - showing quality and status by overriding 53
 - substituting strings 175
 - supported file formats 213
 - types of animations 63–64
 - using anti-aliasing filter 192
 - using buttons 216
 - using custom properties 31
 - viewing in read-only mode 96
 - viewing references to a client control 403
- T**
- tab order 189
 - description 189
 - editing 190
 - TabOrder property 522, 552
 - TabStop property 190, 523, 552
 - tags
 - attribute reference to a SuperTag 278
 - binding animations to ArcestrA attributes 34

- browsing for in a managed application 278
 - connecting to element animations 277–279
 - importing from SmartSymbols 423–424
 - linking to symbol custom properties 258
 - SuperTags 278
 - using values in an ArcestraA symbol 37
 - technical support 18
 - Tension property 523, 540
 - text
 - autoscaling 212
 - drawing boxes 113
 - exporting and importing overridden text
 - with programmatic API 79–83
 - overriding appearance to indicate a change in quality 221
 - overriding appearance to indicate non-good status or quality 197, 224–225
 - placing on the canvas 113
 - scaling size for buttons 216
 - setting Check Box control caption 242
 - setting color 172
 - setting color for trailing dates of Calendar control 249
 - setting color in Windows common controls 239
 - setting default in Edit Box controls 243
 - setting font 172, 173
 - setting text display format 171
 - setting to display 171
 - setting to read-only in Edit Box controls 244
 - setting word wrap in Edit Box controls 243
 - substituting strings 174–175
 - word wrapping 212
 - wrapping for buttons 217
 - Text property 523, 540
 - TextColor property 523, 548
 - TextFormat property 171, 524, 540
 - textures
 - description 185
 - setting 185–186
 - setting transparency 186
 - the 87
 - time
 - Calendar property 240
 - elapsed 256
 - formats for DateTime Picker control 250
 - setting customized in DateTime Picker control 250
 - setting display in DateTime Picker control 250
 - title image display mode 213
 - TitleFillColor property 524, 544
 - TitleTextColor property 524, 549
 - tooltips
 - configuring animations 316–317
 - TopIndex property 524
 - TrailingTextColor property 525, 549
 - translating
 - custom properties 434
 - transparency level 186–187
 - Transparency property 525, 540
 - TransparentColor property 212, 215, 525, 541
 - TreatAsIcon property 526, 553
 - Trend Pen animation 355–359
 - triangular gradient 183
 - truth table fill style animation
 - adding a condition 284
 - changing condition processing order 206, 207, 285
 - configuring element 205–206, 283–284
 - deleting a condition 207, 284
 - setting default 284
 - using default in a truth table 284
 - truth table line style
 - changing condition processing order 288
 - configuring 287
 - deleting a condition 288
 - setting a condition 288
 - setting default properties 287
 - using default properties 288
 - truth table text style animation
 - adding a condition 291
 - changing the condition processing order 291
 - configuring 290
 - deleting a condition 291
 - setting default properties 291
 - using default properties 291
- ## U
- UnFilledColor property 526, 544
 - up image 217
 - UpImage property 217, 526, 541
- ## V
- Value property 239, 526
 - Verified Write 371

- vertical alignment 124
 - vertical gradients 182
 - Vertical pattern property 184
 - vertical slider animations 326
 - VerticalDirection property 527, 545
 - VerticalPercentFill property 527, 545
 - Visible property 143, 527, 553
 - visualization animations
 - description 269
 - hiding 282
 - showing 282
- W**
- While Showing script 376
 - Width property 128, 528, 541
 - WindowMaker
 - deriving an AutomationObject instance by embedding an ArcestrA symbol 22
 - differences between ArcestrA Symbol Editor 55–58
 - marking embedded symbols for an update 42
 - setting time-out period for scripts 375
 - Windows common controls
 - changing color of background and text 239
 - configuring to write data 355
 - data types 240
 - description 238–239
 - drawing 114
 - methods 555–558
 - provided types 28–29
 - reading and writing selected value at run time 239–240
 - reading and writing selected values at run time 239–240
 - types 238
 - value property 239
 - Windows controls
 - drawing 114
 - listing 60
 - WindowViewer
 - viewing languages for symbols 441
 - wizards
 - InTouch 58, 60
 - using 60
 - WordWrap property 217, 528, 541
 - Working with
 - Show/Hide Graphic feature 447
- X**
- X property 122, 529, 542
- Y**
- Y property 122, 529, 542
- Z**
- zoom
 - description 102
 - selected element 100
 - specified area 101
 - specifying value of 101
 - to a specified point 100
 - to default value 100
 - using mouse 102
 - z-order
 - adjusting element order 131
 - changing tab order 189
 - changing within path graphic 159–160
 - linking path graphic elements by 30