Honeywell

Experion
HMIWeb Display Building Guide

EP-DSX164
300
06/06

Release 300

Honeywell

Document	Release	Issue	Date
EP-DSX164	300	1	June 2006

Notice

This document contains Honeywell proprietary information. Information contained herein is to be used solely for the purpose submitted, and no part of this document or its contents shall be reproduced, published, or disclosed to a third party without the express permission of Honeywell Limited Australia.

While this information is presented in good faith and believed to be accurate, Honeywell disclaims the implied warranties of merchantability and fitness for a purpose and makes no express warranties except as may be stated in its written agreement with and for its customer.

In no event is Honeywell liable to anyone for any direct, special, or consequential damages. The information and specifications in this document are subject to change without notice.

Copyright 2006 – Honeywell Limited Australia

Honeywell trademarks

PlantScape®, SafeBrowse®, **TotalPlant**® and TDC 3000® are U.S. registered trademarks of Honeywell International Inc.

Experion™ is a trademark of Honeywell International Inc.

Other trademarks

Microsoft and SQL Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Trademarks that appear in this document are used only to the benefit of the trademark owner, with no intention of trademark infringement.

Support and other contacts

United States and Canada

Contact	Honeywell IAC Solution Support Center
Phone	1-800 822-7673. In Arizona: (602) 313-5558 Calls are answered by dispatcher between 6:00 am and 4:00 pm Mountain Standard Time. Emergency calls outside normal working hours are received by an answering service and returned within one hour.
Facsimile	(602) 313-5476
Mail	Honeywell IS TAC, MS P13 2500 West Union Hills Drive Phoenix, AZ, 85027

Europe

Contact	Honeywell TAC-EMEA
Phone	+32-2-728-2704
Facsimile	+32-2-728-2696
Mail	Honeywell TAC-EMEA Avenue du Bourget, 1 B-1140 Brussels, Belgium

Pacific

Contact	Honeywell Global TAC - Pacific
Phone	1300-300-4822 (toll free within Australia) +61-8-9362-9559 (outside Australia)
Facsimile	+61-8-9362-9169
Mail	Honeywell Global TAC - Pacific 5 Kitchener Way Burswood, WA, 6100, Australia
Email	GTAC@honeywell.com

India

Contact	Honeywell Global TAC - India
Phone	+91-20-2682-2458 / 1600-44-5152
Facsimile	+91-20-2687-8369
Mail	Honeywell Automation India Ltd. 56 and 57, Hadapsar Industrial Estate Hadapsar, Pune -411 013, India
Email	Global-TAC-India@honeywell.com

Korea

Contact	Honeywell Global TAC - Korea
Phone	+82-2-799-6317
Facsimile	+82-2-792-9015
Mail	Honeywell Korea, 17F, Kikje Center B/D, 191, Hangangro-2Ga Yongsan-gu, Seoul, 140-702, Korea
Email	Global-TAC-Korea@honeywell.com

People's Republic of China

Contact	Honeywell Global TAC - China
Phone	+86-10-8458-3280 ext. 361
Mail	Honeywell Tianjin Limited 17 B/F Eagle Plaza 26 Xiaoyhun Road Chaoyang District Beijing 100016, People's Republic of China
Email	Global-TAC-China@honeywell.com

Singapore

Contact	Honeywell Global TAC - South East Asia
Phone	+65-6580-3500
Facsimile	+65-6580-3501 +65-6445-3033
Mail	Honeywell Private Limited Honeywell Building 17, Changi Business Park Central 1 Singapore 486073
Email	GTAC-SEA@honeywell.com

Taiwan

Contact	Honeywell Global TAC - Taiwan
Phone	+886-7-323-5900
Facsimile	+886-7-323-5895 +886-7-322-6915
Mail	Honeywell Taiwan Ltd. 10F-2/366, Po Ai First Rd. Kaohsiung, Taiwan, ROC
Email	Global-TAC-Taiwan@honeywell.com

Japan

Contact	Honeywell Global TAC - Japan
Phone	+81-3-5440-1303
Facsimile	+81-3-5440-1430
Mail	Honeywell K.K 1-14-6 Shibaura Minato-Ku Tokyo 105-0023 Japan
Email	Global-TAC-JapanJA25@honeywell.com

Elsewhere

Call your nearest Honeywell office.

World Wide Web

To access Honeywell Solution Support Online, do the following:

1 In your web browser, type the address `http://www.honeywell.com/ps`.

2 Click **Login to My Account** and then log on.

3 Move the pointer over **Contacts & Support** in the top menu bar and then choose **Support** from the popup menu.

Training classes

Honeywell holds technical training classes on Experion. These classes are taught by experts in the field of process control systems. For more information about these classes, contact your Honeywell representative, or see `http://www.automationcollege.com`.

Related documentation

For a complete list of publications and documents for Experion, see the *Experion Overview.*

Contents

1 Getting started 19

Starting HMIWeb Display Builder 20
Layout of the main window. 21
Toolbars 22
Displaying/hiding toolbars 23
Standard toolbar. 24
Toolbox toolbar 25
Arrange toolbar 28
Transform toolbar 29
View toolbar 30
Format toolbar 31
Drawing toolbar. 32
Scripting toolbar 33
Object Explorer 34
Point Browser 36
Properties Window 38
Shape Gallery 40
Shape library 41

2 Creating displays 43

Creating a new display 44
Creating a standard display 45
Creating a dynamic shape 46
Creating a shape sequence. 48
Creating a popup 49
Creating a faceplate 50
Creating a point detail display. 51
About display types. 53
About dynamic shapes. 54
About shape sequences 55
Using a shape sequence as a status indicator. 55
Using a shape sequence as an animation. 56
About custom properties 57
About popups and faceplates. 59
About display templates 60
About generic displays 61
About archive format. 65

3 Drawing techniques 67

Creating objects . . . 68
Creating a Bezier curve . . . 70
Creating a new object from overlapping objects . . . 72
Selecting objects . . . 74
Selection handles . . . 74
Reference object . . . 75
Deselecting objects . . . 75
Deleting objects . . . 76
Resizing objects . . . 77
Duplicating objects . . . 78
Locking objects . . . 79
Aligning objects . . . 80
Distributing objects . . . 81
Grouping objects . . . 82
Combining objects to improve display performance . . . 83
Flipping objects . . . 84
Rotating objects . . . 85
Defining an object's behaviors . . . 86
Changing the stacking order . . . 87
Using the Node Editor . . . 88
Moving objects with the arrow keys . . . 91
Inserting a graphic . . . 92
About graphic formats . . . 93
Inserting text from another application . . . 95
Inserting a shape sequence . . . 96
Inserting a dynamic shape . . . 97
Embedding a linked shape . . . 98
Upgrading an embedded shape . . . 99
Attaching a popup or faceplate to an object . . . 100
Inserting an ActiveX control . . . 101
Inserting an ActiveX document . . . 102
Defining an object's color . . . 103
Selecting a non-standard color . . . 104
Applying a gradient fill . . . 105
Undoing your work . . . 106

4 Using advanced display features 107

About style sheets . . . 108
Attaching a style sheet to a display . . . 109
Creating a style sheet . . . 111
Structure of a style . . . 112
Referencing style sheets used by shape files . . . 113

Predefined style attributes . . . 114
Special uses for user-defined style attributes . . . 121
Breakpoints . . . 122
About shortcut menus . . . 125
Creating a custom shortcut menu . . . 126
Structure of a shortcut menu . . . 127
Writing scripts for shortcut menus . . . 130
Invoking a shortcut menu from a display script . . . 135

5 Optimizing display performance 137

Using the Display Performance Analysis tool . . . 138
Display-creation guidelines . . . 141
Minimizing the object count . . . 142
Using shapes . . . 144
Creating your own shapes . . . 145
Minimizing the use of behaviors . . . 146
Using scripts . . . 147
Using breakpoints . . . 149
Miscellaneous guidelines . . . 150
Station-related guidelines . . . 151
Improving the performance of an existing display (an example) . . . 152
Improving the design of the dynamic shape . . . 156
Optimizing the performance of shapes containing scripts . . . 158

6 Object properties 161

Animation tab . . . 163
Behaviors tab . . . 165
Standard behaviors . . . 165
Colors tab . . . 167
Custom Properties tab . . . 168
Columns tab (table) . . . 169
Data tab . . . 170
Defining a link to a point . . . 171
Defining a link to a database file . . . 173
Data tab (alarm or event table) . . . 175
Details tab . . . 176
ActiveX document . . . 177
Alarmstate . . . 178
Alarm or event table . . . 179
Alphanumeric . . . 181
.Checkbox . . . 183
Combobox . . . 184
Hyperlink . . . 185
Indicator . . . 186
Picture . . . 187

Pushbutton . . . 188
Shape . . . 190
Table . . . 191
Filters tab . . . 193
Font tab . . . 194
General tab . . . 195
Lines tab . . . 197
Plots tab . . . 198
Popup tab . . . 199
Repeat tab (table) . . . 200
Script Data tab . . . 201
Adding point parameters to an object . . . 203
Scrollbar tab . . . 204
Shortcut Menu tab . . . 207
Sorting tab . . . 208
Period tab . . . 209
View tab . . . 211
Trend . . . 212
Toolbars . . . 213
Plot Area . . . 214
Y Axis . . . 215
X Axis . . . 216
Reference Line . . . 217
Grid Lines . . . 218
Selecting acronyms . . . 219
Specifying URLs . . . 221
About vector graphic objects . . . 222
About trends . . . 223
Chart area . . . 224
Axis . . . 227
Toolbar . . . 228
Legend . . . 229
Scripting trends . . . 230
User interface manipulation . . . 230
Plot manipulation . . . 232
About alarm and event tables . . . 237
Setting alarm states . . . 239
About acronyms . . . 240
About tables . . . 241
Creating a table . . . 241
Modifying a table . . . 242

7 Display/shape properties 243

Appearance tab . . . 244

Callup Task tab . . . 245
Custom Properties tab . . . 246
System Custom Properties tab . . . 247
Shape Custom Properties tab . . . 249
Details tab . . . 250
Defining SafeView window categories . . . 252
General tab . . . 253
Display . . . 254
Shape sequence . . . 255
Dynamic shape . . . 256
Popup . . . 257
Help tab . . . 258
Specifying a topic in a HTML Help file . . . 259
Specifying help other than HTML Help . . . 260
Keys tab . . . 261
Periodic Task tab . . . 262
Shortcut Menu tab . . . 263

8 Setting up HMIWeb Display Builder 265

Setting HMIWeb Display Builder's options . . . 266
General tab . . . 267
Grids and Rulers tab . . . 268
Shapes tab . . . 269
Using the diagnostic and productivity tools . . . 270
Using the HMIWeb Display Builder Assistant . . . 271
Generating a Display Report . . . 272
Renaming a point in the current display . . . 274
Replacing a shape in a display . . . 275
Creating a toolbar . . . 276
Keyboard shortcuts . . . 277
Shortcut menu . . . 278

9 Distributing displays 279

Display storage techniques . . . 280
Distribution (replication) issues . . . 281

10 Display scripting reference 283

11 Scripting basics 285

Scripting languages . . . 286
Setting the default scripting language . . . 286
Events . . . 287
Event firing order . . . 288
General section . . . 290
Timers . . . 291

Referencing objects . . . 293
Working with shapes . . . 296
Working with popups and faceplates . . . 298
Accessing Station-related properties in page-level scripts . . . 300
Station runtime status . . . 301
Setting and returning colors . . . 303
Reading and writing to point parameters . . . 304
Properties and methods of points on the Script Data tab . . . 306
Debugging scripts that access points . . . 308
Using custom properties in scripts . . . 309
Accessing the point mapped to a shape custom property . . . 310
Accessing the point mapped to a system custom property . . . 312
Re-mapping a shape custom property to another point at run time . . . 314
Re-mapping a system custom property to another point at run time . . . 315
Using a shape custom property to store a user-defined value . . . 317
Using a system custom property to store a user-defined value . . . 318
Scripting in a multi-window environment . . . 319
Transferring data between windows . . . 319
Writing Station-level scripts . . . 320
Using Station Scripting Objects . . . 320
Using Script Editor in Station . . . 320
Specifying dimensions and position . . . 321
Comparing strings . . . 322
Using error handlers . . . 323
Scripting differences between HMIWeb and DSP displays . . . 324
Useful Web sites and reference books . . . 326

12 Using the Script Editor 327

Using the Script Editor in Station . . . 328
Writing Scripts for Station . . . 328
Activating Intellisense . . . 329
Writing scripts . . . 330
Finding/replacing strings in scripts . . . 331

13 HMIWeb Object Model reference 333

Overview of the HMIWeb Object Model . . . 334
Overview of the trend object model . . . 336
Objects . . . 337
ActiveX controls . . . 338
Alphanumeric object . . . 339
Application object . . . 342
AppWindow object . . . 350
ChartArea object . . . 352
ChartToolbar object . . . 354

Checkbox object . . . 355
Combobox object . . . 357
Dictionary object . . . 360
DisplayDataRepository object . . . 361
event object . . . 362
EventBar object . . . 364
Group object . . . 366
Hyperlink object . . . 368
Indicator object . . . 370
Legacy object . . . 372
Legend object . . . 375
Node object . . . 377
Nodes object . . . 378
NumericHistory object . . . 379
Page object . . . 380
Picture object . . . 382
Plot object . . . 384
Plots object . . . 386
PlotCanvas object . . . 387
PlotDataTip object . . . 389
Popup object . . . 390
Pushbutton object . . . 391
ReferenceCursor object . . . 393
RuntimeStatus object . . . 394
Shape object . . . 398
Station object . . . 401
StationDataRepository object . . . 404
StationWindows object . . . 405
style object . . . 407
Text object . . . 408
TitleToolbar object . . . 410
Trend object . . . 411
Vector Graphic object . . . 413
window object . . . 416
XAxis object . . . 417
YAxis object . . . 419
Methods . . . 421
Add method . . . 422
AddNewPlot method . . . 423
AddPlotData method . . . 425
AddString method . . . 427
Autoscale method . . . 428
Blue method . . . 429
blur method . . . 430
CancelChange method . . . 431
CancelEvent method . . . 432

clearInterval method 433
CancelOperatorCommand method 434
CancelResponse method 435
ClearPlotData method 436
clearTimeout method 437
click method 438
Connect method 439
Copy method 440
CopyDataToClipboard method 441
CreatePopupWindow method 442
CreateTimer method 443
DecreaseZoomLevel 444
DeleteString method 445
DisplayExists method 446
DPToLPX and DPToLPY methods 447
ExecuteOperatorCommand method 448
FindFile method 449
FindPopup method 450
focus method 451
FlagUserInteraction method 452
GetConnectionProperty method 453
GetCustomProperty method 454
GetDataTipArrowColors method 455
GetDataTipFormat method 456
GetDataTipVisible method 457
GetGridLinePen method 458
GetInPlaceEditing method 459
GetItemVisible method 460
GetLabels method 461
GetLineStyle method 462
GetPen method 463
GetPeriod method 464
GetPlotColor method 465
GetPlotDiscrete method 466
GetPlotMarkerStyle method 467
GetPlotStepMode method 468
GetPlotVisible method 469
GetRange method 470
GetReferenceValues method 471
GetResponse method 473
GetResponse2 method 475
GetResponse3 method 477
GetResponse4 method 479
GetSampleInterval method 480
GetStyleProperty method 481
GetStyleClassProperty method 483

GetValue method 484
GetXAxisToolbarItemVisible method 485
GetXAxisScale method 486
GetYAxisScale method 487
GetXAxisUnzoomedScale 488
GetYAxisUnzoomedScale 489
GetZoomRegion method 490
Green method 491
HideAllDataTips method 492
HideDataTips method 493
IncreaseZoomLevel method 494
InvokeCommand method 495
InvokeMenu2 method 496
InvokePopup method 497
Item method 498
KillTimer method 500
LogMessage method 501
LPToDPX and LPToDPY methods 502
MakeColor method 503
MessageBox method 504
Objects method 505
PutValue method 506
Quit method 507
RaiseCustomEvent method 508
Red method 509
Redraw method 510
Refresh method 511
ReloadContent method 512
Remove method 513
RemovePlot method 514
RemovePlotData method 515
RequestDemandData method 516
RequestServerLRN method 518
RequestTask method 519
ResetContent method 521
RuntimeDataRequest method 522
Scroll method 523
SendKeyPress method 524
SetCommandUIState method 525
SetCustomProperty method 526
SetDataTipArrowColors method 527
SetDataTipFormat method 528
SetDataTipVisible method 529
SetGridLinePen method 530
SetInPlaceEditing method 531
setInterval method 532

SetItemVisible method . . . 533
SetLabels method . . . 534
SetLineStyle method . . . 535
SetNodePoint method . . . 536
SetPen method . . . 537
SetPeriod method . . . 538
SetPlotColor method . . . 539
SetPlotDiscrete method . . . 540
SetPlotMarkerStyle method . . . 541
SetPlotPen method . . . 542
SetPlotRange method . . . 543
SetPlotStepMode method . . . 545
SetPlotVisible method . . . 546
SetSampleInterval method . . . 547
SetSampleIntervaltoFastHistory method . . . 548
setTimeout method . . . 549
SetTimeRange method . . . 550
SetTitleBarFormat method . . . 551
SetUserVersionInfo method . . . 553
SetXAxisToolbarItemVisible method . . . 554
SetXAxisScale method . . . 555
SetYAxisScale method . . . 556
SetXAxisUnzoomedScale method . . . 557
SetYAxisUnzoomedScale method . . . 558
SetZoomRegion method . . . 559
Shell method . . . 560
UnZoom method . . . 561
UserObjectNotify method . . . 562
Events . . . 563
onactivate event . . . 564
OnActivated event . . . 565
OnAlarm event . . . 566
OnAppStartup event . . . 567
OnAppStatusUpdate event . . . 568
OnBeforeConnect event . . . 569
OnBeforeOKBKeyPressed event . . . 570
OnBlink event . . . 571
onblur event . . . 572
onchange event . . . 573
onclick event . . . 574
OnConnect event . . . 575
oncontextmenu event . . . 576
OnCustomEvent event . . . 577
ondatachange event . . . 579
ondblclick event . . . 581
ondeactivate event . . . 582

OnDictionaryValueChanged event 583
OnDisconnect event. 584
OnDisplayNotFound event 585
OnEventDataTipDisplayed event 587
onfocus event. 588
onkeydown event. 589
onkeypress event 590
onkeyup event 591
OnMenu event 592
OnMenu2 event 593
onmousedown event 594
onmouseenter event 595
onmouseleave event. 596
onmousemove event 597
onmouseout event 598
onmouseover event 599
onmouseup event 600
OnOKBKeyPressed event 601
OnOperatorChange event 602
OnOperatorCommand event 603
OnOperatorCommand2 event 604
onoperatorkey event 606
onpagecomplete event 611
OnPageLoad event. 612
OnPause event 613
OnPeriodicUpdate event 614
OnPlotDataBufferFull event 615
OnPlotDetailsChanged event. 616
OnQuit event 617
OnReferenceCursorSet 618
OnRequestDataTip event. 619
OnRequestEventInfo event 620
OnRequestReferenceDataTip event 621
OnResponse event 622
OnResume 623
OnScrollXAxis 624
OnScrollYAxis event 625
onshapeload event 626
onshapeunload event 627
OnSilenceIKB event 628
OnTimer event. 629
OnTimSelectorPositionChanged event 630
OnTrendLoadComplete event 631
onunload event. 632
onupdate event. 633
OnUserObjectNotify event 634

OnUnZoomed event 635
OnZoom event 636
Legacy_OnChartLoadComplete event 637
Legacy_OnChartUnZoomed event 638
Legacy_OnChartZoom event 639
Legacy_OnChartScroll event 640
Legacy_OnDataBufferFull 641
Legacy_OnEventBoxDisplayed event 642
Legacy_OnEventDetailDisplayed event 643
Legacy_OnReferenceLineSet event 644

14 Example scenarios 645

Copying example scripts 647
Calling up a faceplate on an alarm condition 648
Changing a ToolTip based on the state of a pump 649
Creating an animation with a shape sequence 650
Using timers in a shape file 652
Creating a moving object 654
Controlling the color of an object in a dynamic shape 656
Using color to reflect the value of a status point 658
Using a button to start a pump 659
Sending a display to another Station in a console 661
Creating a jogging control 662
Performing an action in response to a right-click 663
Checking the user's security level 664
Starting an application by clicking a button 666
Controlling Station from another application 667
Responding to Station events from another application 668
Forcing a display to use a particular SafeView window 670
Changing a trend's contents at run time 671
Showing and hiding objects 673
Using event bubbling to reduce script effort 674

Getting started

1

HMIWeb Display Builder is a specialized drawing application that enables you to create your own (*custom*) displays for Station.

Custom displays allow you to present information in a sophisticated and user-friendly manner—well-designed custom displays make it easier for operators to visualize complex processes, and reduce the probability of operator errors.

Attention

This version of Display Builder creates Web-based displays—called *HMIWeb displays*. If you want to create displays that use the proprietary DSP format, see the *Display Building Guide*.

Starting HMIWeb Display Builder

To start HMIWeb Display Builder on the server:

1 From the **Start** menu, choose **Start** > **Programs** > **Honeywell Experion PKS > Server** > **HMIWeb Display Builder**.

To start HMIWeb Display Builder on a client computer:

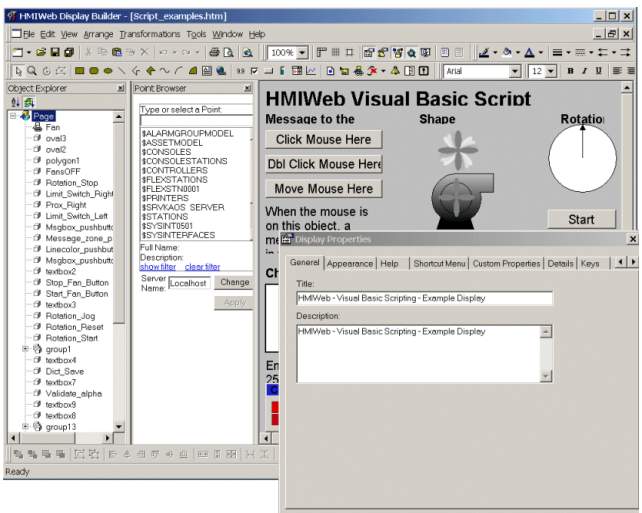
1 From the **Start** menu, choose **Start** > **Programs** > **Honeywell Experion PKS > Client Software** > **HMIWeb Display Builder**.

Layout of the main window

The following figure shows a typical layout, in which:

- The display(s) you are editing appears on the right
- The Object Explorer, which lists all objects in the display, appears on the left
- The Point Browser, which lists all of the points on the server to which you are connected, appears on the left next to the Object Explorer.
- The toolbars appear at the top and bottom of the window
- The “floating” Properties Window shows the properties (characteristics) of the selected object

Figure 1 Typical layout



Toolbars

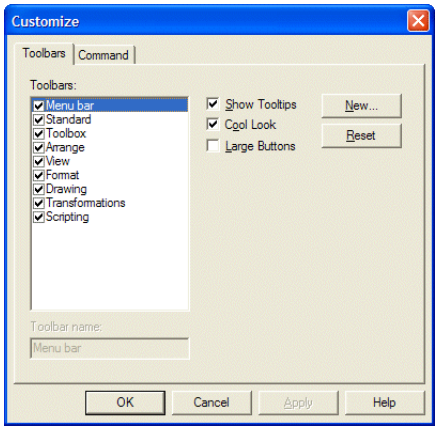
The toolbars provide speedy access to many functions.

If you want to create your own toolbar, see “Creating a toolbar” on page 276.

Displaying/hiding toolbars

To display (or hide) a toolbar:

1 Choose **Tools** > **Customize** and click the **Toolbars** tab to see the list of toolbars.



A check mark opposite a toolbar indicates that it is already visible.

2 Select the toolbar you want to display (or hide).

3 If necessary, drag the toolbar to a convenient location. (If you drag a toolbar to the top or bottom of the window, it will "dock" to the window's border.)

Standard toolbar

The Standard toolbar includes basic Windows-related buttons such as **Open** and **Save**.

Button	Description
	New Display. Creates a new display. Click the arrow to the right of the button to display the list of display types. See “About display types” on page 53.
	Open. Opens an existing display.
 	Save. Saves the current display. **Save All**. Saves all open displays.
	Cut. Removes the selected object(s) from the display (and copies them to the clipboard).
	Copy. Copies the selected object(s) to the clipboard.
	Paste. Pastes the clipboard’s contents into the display.
	Duplicate. Makes duplicates of the selected object(s).
	Delete. Deletes the selected object(s).
 	Undo. Undoes changes you have made to the display. **Redo**. Redoes changes you have undone. See “Undoing your work” on page 106.
	Print. Prints the display.
	Print Preview. Shows the display as it will print. You can then either print the display by clicking **Print**, or return to editing mode by clicking **Close**.
	Preview. Shows the display as it will appear in Station. Preview also allows you to test scripts, providing they do not require interaction with the server. For example, clicking an object will run the object’s onclick script. Click **Close** to return to the normal editing mode.

Toolbox toolbar

The Toolbox toolbar contains two types of buttons:

- Editing buttons (used to edit existing objects)
- Object-creation buttons (used to create objects)

Editing buttons

Button	Description
	Pointer. The main tool you use to select, move and resize objects. See “Selecting objects” on page 74.
	Zoomer. Zooms in and out of the display. To zoom in on a part of the display, first click **Zoomer** and then click the area of interest. You can zoom in further by clicking repeatedly. To zoom out, hold down SHIFT and click.
	Rotator. Rotates the selected object(s). See “Rotating objects” on page 85.
	Node Editor. Allows you to change the shape of an object by moving or adding individual *nodes* (apexes or reference points). See “Using the Node Editor” on page 88.

Object-creation buttons

To see how to use these buttons, see “Creating objects” on page 68.

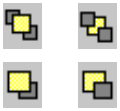
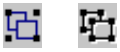
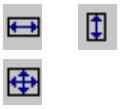
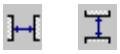
Button	Description
	Rectangle. Creates a rectangle or square.
	Rounded Rectangle. Creates a rectangle or square with rounded corners.
	Oval. Creates an oval or circle.
	Line. Creates a straight line.
	Polyline. Creates a multi-segment line (which is similar to a polygon, but with one open side). To create a polyline, click to mark each node, except for the last. You mark the last node by double-clicking.

Button	Description
	Polygon. Creates a polygon. To create a polygon, you click to mark each node, except for the last. You mark the last node by double-clicking, which then closes the polygon.
	Bezier Curve. Creates a smooth-curved line. See “Using the Node Editor” on page 88.
	Arc. Creates an arc (a quarter of an oval or circle).
	Wedge. Creates a segment (quarter) of an oval or a circle.
	Textbox. Creates block of text.
	Hyperlink. Creates a hyperlink. When a user clicks the hyperlink, Station calls up the specified URL (such as a Web page) or display.
	Alphanumeric. Used to display database values in many different formats.
	Check box. Creates a check box so that users can select or deselect an option in an interactive display.
	Pushbutton. Creates a button that users can click to perform a specified command.
	Indicator. Creates a gauge-like object that shows a relative value (similar in principal to a car’s fuel gauge).
	Combobox. Creates a combobox so that users can select from a list of options in an interactive display.
	Trend. Creates a trend (graph) that shows changes in values over time. See “About trends” on page 223.
	ActiveX Document. Inserts a link to an ActiveX document, such as a Word document, into the display. See “Inserting an ActiveX document” on page 102.
	Picture. Inserts a picture (graphic). See “Inserting a graphic” on page 92.
	Shape. Inserts a shape sequence or dynamic shape. See “Inserting a shape sequence” on page 96 and “Inserting a dynamic shape” on page 97.
	ActiveX Control. Inserts a link to an ActiveX Control. See “Inserting an ActiveX control” on page 101.
	Alarm Table. Creates a specialized table that lists alarms. See “About alarm and event tables” on page 237.

Button	Description
	Event Table. Creates a specialized table that lists events. See “About alarm and event tables” on page 237.
	Alarm State. Inserts an alarm icon (as used in the Alarm Summary) which indicates an alarm state.
	Table. Inserts a table into a display. See “About tables” on page 241.

Arrange toolbar

The Arrange toolbar contains buttons for aligning and grouping objects.

Button	Description
	These buttons change the order in which objects are *stacked* on top of each other. See “Changing the stacking order” on page 87.
	These buttons group/ungroup objects. See “Grouping objects” on page 82.
	These buttons align objects. See “Aligning objects” on page 80.
	These buttons change the size of objects so that they are the same size as a reference object. See “Resizing objects” on page 77.
	These buttons move objects so that they are evenly spaced. See “Distributing objects” on page 81.
	Locks/unlocks an object. Locking an object protects it from being accidently moved or resized. See “Locking objects” on page 79.

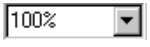
Transform toolbar

The Transform toolbar contains buttons for transforming objects.

Button	Description
	Flip Horizontal, **Flip Vertical**. These buttons perform mirror-image transformations on objects. See “Flipping objects” on page 84.
	Rotate Left, **Rotate Right**. These buttons rotate objects +/-90 degrees. See “Rotating objects” on page 85.
	Union, **Difference**, **Intersection**, **Exclusive OR**, **Join**. These buttons create a new object from two or more overlapping objects. See “Creating a new object from overlapping objects” on page 72.
	Convert to Path. Converts objects, such as rectangles and circles, to *paths* (lines and curves). Having converted an object to a path, you can edit its nodes—see “Using the Node Editor” on page 88.
	Combine, **Un-Combine**. These buttons combine/uncombine static objects, such as lines and rectangles, which helps improve display performance. See “Combining objects to improve display performance” on page 83.

View toolbar

The View toolbar includes a range of editing buttons.

Button	Description
	**Zoom**. Sets the "magnification" for the display. Use the **Zoomer**— —to zoom in on a particular part of the display.
	Rulers. Shows/hides the rulers. The rulers make it easier to arrange objects in a neat and consistent manner.
	Grid. Shows/hides the grid. The grid makes it easier to arrange objects in a neat and consistent manner. (To set the grid spacing and color, see "Setting HMIWeb Display Builder's options" on page 266.)
	Snap-to-Grid. Turns the "snap-to-grid" function on/off. When snap-to-grid is on, objects automatically align to the grid when you create, move or resize them.
	Properties Window. Opens/closes the "Properties Window" on page 38, which you use to edit the properties of the selected object(s).
	Script Editor. Opens/closes the Script Editor, which you use to write *scripts* (a script is a mini-program that performs a specific task). See "Using the Script Editor" on page 327.
	Object Explorer. Shows/hides the Object Explorer, which lists every object in the display. See "Object Explorer" on page 34.
	Point Browser. Shows/hides the Point Browser, which lists points on a selected Experion server. See "Point Browser" on page 36.
	Shape Gallery. Shows/hides the Shape Gallery, which makes it easy to preview and add shapes to your displays. See "Shape Gallery" on page 40.
	HTML Source. Opens the HTML source for the display in a text editor. If you are familiar with HTML, you can change the display by editing the HTML source.
	Style Sheet. If a style sheet is attached to the display, opens it in a text editor. To learn about style sheets, see "About style sheets" on page 108.

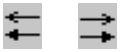
Format toolbar

The Format toolbar contains buttons for controlling the appearance of text.

Button	Description
[Default] 12pt	**Font**. Sets the font of text. Note that you should only use fonts that are loaded on every Station computer. **Font Size**. Sets the size of text. (If you want to specify a non-standard size, click the box and type the size—for example: **`56.5pt`**—and then press **ENTER**.)
B *I* <u>U</u>	**Bold**, **Italic** and **Underline**. Set the text's appearance.
	Align Left, **Align Center**, **Align Right** and **Justify**. Set the text alignment within a paragraph.

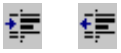
Drawing toolbar

The Drawing toolbar contains buttons for controlling colors and line properties.

Button	Description
	Line Color. Sets the line color of objects. Click the button to apply the default color, or click the arrow to the right of the button to select another color—see “Defining an object’s color” on page 103.
	Fill Color. Sets the fill color of objects. Click the button to apply the default color, or click the arrow to the right of the button to select another color—see “Defining an object’s color” on page 103.
	Text Color. Sets the fill color of text. Click the button to apply the default color, or click the arrow to the right of the button to select another color—see “Defining an object’s color” on page 103.
	Line Width. Sets the line thickness.
	Line Style. Sets the line style, such as solid, dotted or dashed.
	**Start Arrow** and **End Arrow**. Set the line arrow style.

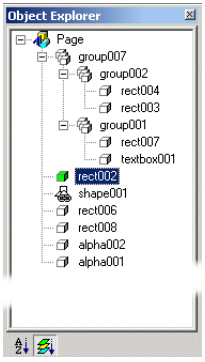
Scripting toolbar

The Scripting toolbar is used in conjunction with the Script Editor to write scripts. (See Using the Script Editor.)

Button	Description
	**Indent**, **Outdent**. Increase/decrease the indent of the selected line.

Object Explorer

The Object Explorer is a powerful tool for managing and editing objects. It shows every object in the current display and, in the case of group objects, shows the object hierarchy.



If the Object Explorer is not visible, click on the View toolbar.

Using the Object Explorer

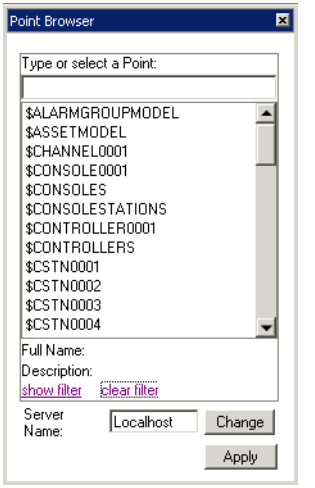
The following table describes the tasks you can perform in the Object Explorer.

To:	Do this:
Select one object	Click it.
Select several objects	Hold down CTRL and click each object.
Select a range of objects from the list	Hold down SHIFT and click the first and last objects you want to select.
Edit the properties of one or more objects	Select the objects as described above and edit the properties in the Properties Window.
Sort objects in alphabetical order	Click at the bottom of the Object Explorer.
Sort objects in their *stacking* order (the order in which objects are stacked on top of each other)	Click at the bottom of the Object Explorer. By default, the stacking order is the same as the order in which you created the objects—that is, the first object you created is at the bottom of the stack and the last object is on the top. (You can manually change the stacking order—see “Changing the stacking order” on page 87.)

To:	Do this:
Show/hide objects within a group	Click the “+” or “-” to the left of the group icon. (Alternatively, double-click the icon itself.)
Copy an object to the same or another display	Drag the object from the Object Explorer and drop it on the other display. (Both displays need to be visible to do this.)

Point Browser

The Point Browser displays every point that has been defined on the server to which you are connected.



Using the Point Browser

The following table describes the tasks you can perform using the Point Browser.

To:	Do this:
Select a point	Click it.
Filter the list of points	Click Show Filter. The enterprise model appears. Expand the model by which you want to filter and click the appropriate entity in the model. The point list changes to show only those points that belong to the entity.
Remove the filter	Click **Clear Filter**
Connect to a different server	Click the **Change** button. Select the server from the list of servers and click **OK**.
Include a point in display	Click on the point name in the Point Browser and then drag and drop the point on the display.

To:	Do this:
Apply a point to a display element	Choose one of the following: 1. Drag the point name from the point browser onto the "point" field of the data property page of the display element. 2. Select the display element then select the point in the point browser and click **Apply**.

Properties Window

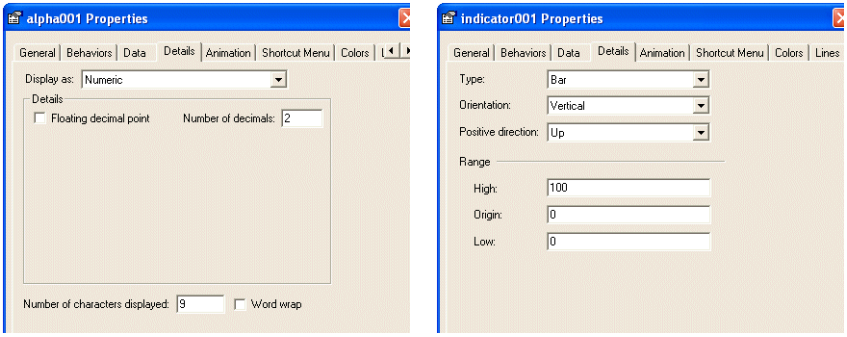
You use the Properties Window to view and edit the *properties* of the selected objects—the properties include physical properties, such as color and size, as well as database link details.

You also use the Properties Window to view and edit the display's properties—these appear when no objects are selected.

If the Properties Window is not visible, click on the View toolbar.

The Properties Window changes whenever you select another object. In the following figure, the left-hand example shows the window when you select an alphanumeric, whereas the right-hand example shows the window when you select an indicator.

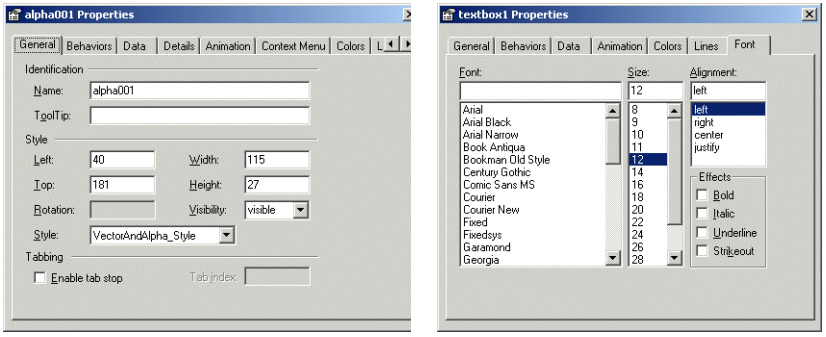
Figure 2 Selecting an object changes the Properties Window



The Properties Window is *modeless*, which means that the selected object(s) change as soon as you change a property's value. (Unlike a standard dialog box, you do not have to click an **OK** button.)

Properties are grouped according to the tabs that run across the top of the window. The following examples show how the window changes as you click the tabs.

Figure 3 Properties are grouped according to tab



The Properties Window also changes if you:

- Select several objects of the same type. Any properties whose values are not identical are blanked or grayed out. For example, **Name** on the **General** tab will be blank because each object has a unique name.

  If you edit a property, you change all selected objects—for example, if you change **Fill Color** to red, they will all turn red.

- Select objects of various types, such as a pushbutton and an alphanumeric. In this case, the window only displays tabs that are shared by all selected objects. For example, if you selected an alphanumeric and a pushbutton, you could change the properties on the **Font** tab because both objects contain text.

- Deselect all objects. The Properties Window then shows the display's properties, and its title changes to "Display Properties".

For a description of each property, see:

- "Object properties" on page 161
- "Display/shape properties" on page 243

Shape Gallery

The Shape Gallery makes it easy to preview and insert shape sequences and dynamic shapes into your displays.

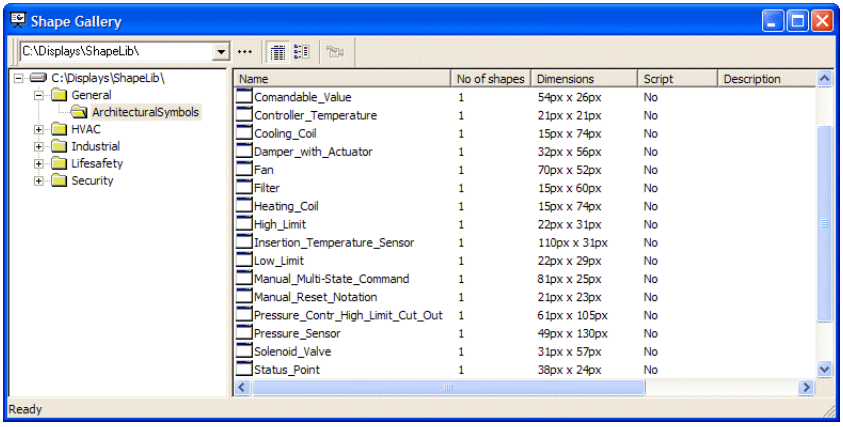
If the Shape Gallery is not visible, click on the View toolbar.

Notes

- You cannot insert a linked shape unless you have registered the folder that contains the shape. See “Setting HMIWeb Display Builder’s options” on page 266.

To preview shape sequences/dynamic shapes:

1 Select the appropriate folder in the left-hand pane. A one-line summary of each file appears in the right-hand pane.



2 If you want to preview each shape, click the button.

3 In you want to preview individual shapes within a shape sequence (where **No of shapes** is 2 or more), double-click it. (Alternatively, select the shape and click the button.)

To insert a shape sequence/dynamic shape into a display, either:

- Drag it from the list and drop it into the display.
- Right-click it and choose **Insert into Display** from the shortcut menu.

Shape library

HMIWeb Display Builder is supplied with a library of pre-built shapes that you can include in your displays. These shapes are stored in `\ShapeLib`, in subject-based folders. (The `\Examples` folder contains example displays that use shapes from the shape library.)

Creating displays

2

This chapter describes the various types of display, and how to create displays.

Creating a new display

Notes

- You cannot change the display's type after creating it. For example, you cannot change a display into a shape sequence.
- An asterisk (*) appears in the title bar, next to the display's filename when you have changed the display, but not saved those changes.
- If you need to keep track of changes to displays, you should save them in *archive format* and use a version management tool such as Microsoft Visual Source Safe. (To learn about archive format, see "About archive format" on page 65.)
- The default folder for custom displays is `Program Files\Honeywell\Experion PKS\Client\Abstract`.

To create a:	Go to:
Display	page 45
Display from a Web page	page 45
Dynamic shape	page 46
Shape sequence	page 48
Popup	page 49
Faceplate	page 50
Point detail display	page 51

Creating a standard display

Notes

- To speed up your work and to ensure visual consistency, you should create an appropriate set of templates and style sheets. See “About display templates” on page 60 and “About style sheets” on page 108.

To create a display:

1. Click the arrow to the right of and choose **Display from Template**.
2. Select the appropriate template and click **OK**.

   A new display, based on the template, appears.
3. Configure the display’s properties as appropriate—see “Display/shape properties” on page 243.
4. Add objects to the display as required—see “Creating objects” on page 68.

Creating a display from a Web page

You can create a display that is based on a Web page. For example, you may want to use a modified version of your intranet’s home page as Station’s default display.

Notes

- If you use the zoom function in HMIWeb Display Builder or Station, scrollbars are not correctly resized.

To create a display from a Web page:

1. Choose **File** > **Open** and select the Web page you want to use.

   The Web page opens.
2. Choose **File** > **Save As** and give it a suitable name.

   It is a now a display—which means you can configure its properties and add HMIWeb objects such as buttons and indicators.

Creating a dynamic shape

Notes

- To learn about dynamic shapes, see “About dynamic shapes” on page 54.
- Before creating a dynamic shape, you should define the purpose of each:
  - Dynamic object you intend to include in the shape.
  - Custom property you intend to include in the shape. You should also give each custom property a name that reflects its purpose. For example, if a custom property represents pressure, you could name it “Pressure”. (For an introduction to custom properties, see “About custom properties” on page 57.)
- All objects within a dynamic shape must be grouped to form one object.
- You should store the shape in a registered folder. See “Setting HMIWeb Display Builder’s options” on page 266.
- To learn how to insert a dynamic shape into a display, see “Inserting a dynamic shape” on page 97.

To create a dynamic shape:

1. Click the arrow to the right of and choose **Dynamic Shape**.

   A blank dynamic shape appears.
2. Open the Properties Window, click the Custom Properties and define each custom property. See “Shape Custom Properties tab” on page 249.
3. Add objects to the shape as required. See “Creating objects” on page 68.
4. Assign custom properties to the dynamic objects, as appropriate. See “Assigning shape custom properties to dynamic objects” on page 46.
5. Group all objects into one group before closing (and saving) the dynamic shape.

Assigning shape custom properties to dynamic objects

You must assign the appropriate shape custom properties to the dynamic object(s) within the shape. (When you insert the shape into a display, the custom properties enable you to map the dynamic objects to points/parameters.)

Attention

You cannot directly assign system custom properties to the dynamic objects contained within a shape. However, after a shape has been inserted into a display it is possible to map the custom properties of the shape to system custom properties (see “Using system custom properties to map objects to points” on page 58).

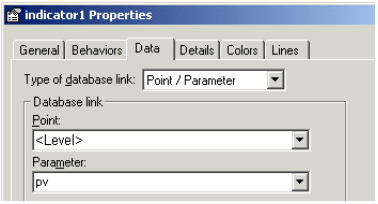
To assign the custom properties:

1. Select a dynamic object and click the **Data** tab the Properties Window. (Select the **Details** tab if the object does not have a **Data** tab.)
2. From the **Point** list select the appropriate custom property.

   Note that custom properties are enclosed within angle brackets (< >).
3. From the **Parameter** list select the appropriate parameter or custom property.
4. Repeat steps 1 and 2 for the remaining dynamic objects that you want to map to shape custom properties.

Example

The following figure shows how to assign the custom properties for the example described in “About custom properties” on page 57.



Creating a shape sequence

Notes

- To learn about shape sequences, see “About shape sequences” on page 55.
- The following rules and restrictions apply to shape sequences:
  - Each shape must be a single object or group.
  - Shapes must be arranged (left-to-right/top-to-bottom) in the appropriate order. That is, the first shape is the top-left one, and so on.
  - If the shape sequence is to be used as a status indicator, you can use the first shape to represent a “bad value”. This shape is only displayed when the value cannot be determined—because, for example, the server cannot access the associated field device. When creating a shape sequence file, you must select **Use first shape for bad value** on the **General** tab.
  - If the shape sequence is to be used as a status indicator for a status point, the number of shapes must be greater than or equal to the number of parameter states. (Plus an extra shape if you want a “bad value” shape.)

    If the shape sequence has more shapes than the parameter has states, the remaining shapes are never used. For example, if a parameter has four states and the shape sequence has nine shapes (including a “bad value” shape), only shapes 2 to 5 shapes are used.
  - A shape can include dynamic objects. However, for performance reasons, you should minimize the use of such objects and thoroughly check the shape sequence’s performance before issuing it.
- If you intend to add custom properties to the shape, you should give them names that reflect their purpose. For example, if a custom property represents pressure, you could name it “Pressure”. (For an introduction to custom properties, see “About custom properties” on page 57.)
- You should store the shape sequence in a registered folder. See “Setting HMIWeb Display Builder’s options” on page 266.
- To learn how to insert a shape sequence into a display, see “Inserting a shape sequence” on page 96.

To create a shape sequence:

1. Click the arrow to the right of and choose **Shape Sequence**.

   A blank shape sequence appears.
2. Configure the shape sequence’s properties as appropriate—see “Display/ shape properties” on page 243.
3. Add shapes (objects) as required—see “Creating objects” on page 68.

Creating a popup

Notes

- To learn about popups, see “About popups and faceplates” on page 59.

To create a popup:

1. Click the arrow to the right of and choose **Popup**.

   A blank popup appears.
2. If required, change the popup’s size on the **Appearance** tab on the Properties Window. (The dotted lines indicate the default size, which is 300x300 pixels.)
3. Add objects to the popup as required—see “Creating objects” on page 68.

Creating a faceplate

Notes

- To learn about faceplates, see “About popups and faceplates” on page 59.
- The filename of the faceplate must be the same as the matching group faceplate template display (or point detail display), except for the “_fp” suffix. For example, if the filename of the point detail display for a particular type of container point is `BoilerType1.htm`, the filename of the faceplate must be `BoilerType1_fp.htm`.

To create a faceplate:

1 Click the arrow to the right of and choose **Popup**.

A blank faceplate appears.

2 If required, change the faceplate’s size on the **Appearance** tab on the Properties Window. (The dotted lines indicate the default size, which is 300x300 pixels.)

3 For each parameter you want users to see:

a. Add an appropriate object—typically, an alphanumeric or indicator.

b. On the **Data** tab of the Properties Window, configure the properties as follows. (Select the **Details** tab if the object does not have a **Data** tab.)

Property	Setting
Type of database link	`Point/Parameter`
Point	`CurrentPoint`
Parameter	The parameter this object represents. For example, `PV` or `SP`.

4 Add other objects, such as a background graphic, as required.

5 Save the faceplate with the appropriate filename (including the “`_fp`” suffix).

Creating a point detail display

A point detail display is used to display point information for all points of a particular type. (Experion is supplied with point detail displays for common point types.)

Each time you create a new type of process, flexible or container point, you must create a matching point detail display. You then add an appropriate object—such as an alphanumeric or indicator—for each parameter you want users to see.

The following procedure summarizes the special steps required to create a point detail display.

To create a point detail display:

1. Click the arrow to the right of and choose **Display from Template**.
2. Select the `Point Detail` template and click **OK**.

   A new display, based on the template, appears.
3. Add an appropriate object (typically an alphanumeric or indicator) for each parameter you want users to see.
4. Open the Properties Window.
5. Select each of these objects in turn and specify the parameter's details. (For most objects, use the **Data** tab; for a chart, use the **Details** tab.)

   The parameter address syntax depends on the point type:

   - Process point parameter
   - Flexible point parameter
   - Container point parameter

Process point parameter

Property	Description
Type of database link*	Select `Point/Parameter`.
Point	Select `<CurrentPoint>`.
Parameter	The syntax is: *`ContainedObject.Parameter`* Where: • *`ContainedObject`* is the contained CM within the point, or a basic block (such as a PID) • *`Parameter`* is the name of the parameter within the contained block

Property	Description
Data entry allowed*	Allows users to change the parameter's value. If you allow data entry, select the minimum **Security Level** required to change the value.

* Not applicable to a chart.

Flexible point parameter

Property	Description
Type of database link*	Select `Point/Parameter`.
Point	Select `<CurrentPoint>`.
Parameter	The parameter's name.
Data entry allowed*	Allows users to change the parameter's value. If you allow data entry, select the minimum **Security Level** required to change the value.

* Not applicable to a chart.

Container point parameter

Property	Description
Type of database link*	Select `Point/Parameter`.
Point	Select `<CurrentPoint>`.
Parameter	The syntax is: *`Alias.Parameter`* Where: • *`Alias`* is the name you defined in Quick Builder for the contained (child) point • *`Parameter`* is parameter's name. For example, if you wanted to display the PV of a contained point whose alias is "HoursRun", you would type: **`HoursRun.PV`**
Data entry allowed*	Allows users to change the parameter's value. If you allow data entry, select the minimum **Security Level** required to change the value.

* Not applicable to a chart.

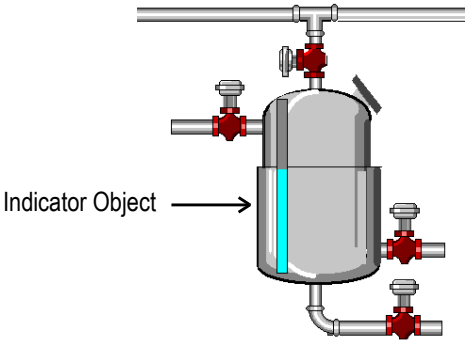
About display types

Display type	Description
Display	A standard/non-specialized display.
Dynamic Shape	A *dynamic shape* is, in effect, a “custom object” that is used in displays to present complex dynamic data. See “About dynamic shapes” on page 54.
Shape Sequence	A *shape sequence* is, in effect, a “custom object” that is used in displays as a status indicator or an animation. See “About shape sequences” on page 55.
Popup	A popup is a secondary window that appears when a user clicks the object to which the popup is attached. See “About popups and faceplates” on page 59.

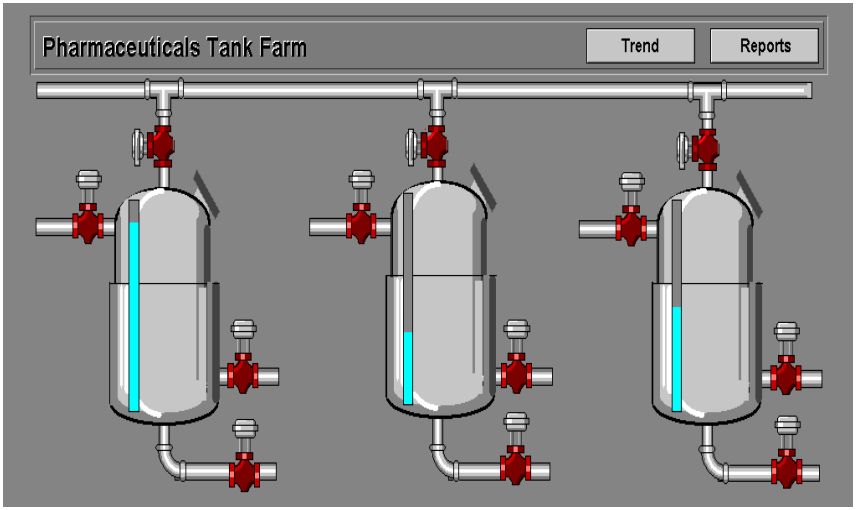
About dynamic shapes

A *dynamic shape* is, in effect, a “custom object” that is used in displays to present complex dynamic data.

The following figure shows a typical dynamic shape that represents a tank. When used in a display, the dynamic object—the indicator—is linked to tank-related parameters such as the liquid level.



The following figure shows a display that contains three copies of the dynamic shape. Each copy is configured to show the level for a particular tank.



About shape sequences

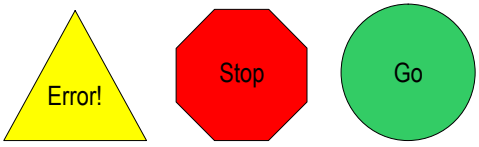
A *shape sequence* is, in effect, a "custom object" that is used in displays as either a status indicator or an animation.

Notes

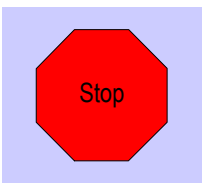
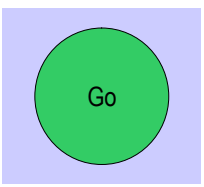
- To learn how to create a shape sequence, see "Creating a shape sequence" on page 48.
- To learn how to insert a shape sequence into a display, see "Inserting a shape sequence" on page 96.

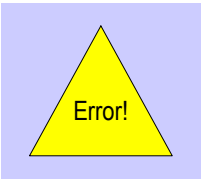
Using a shape sequence as a status indicator

The following figure shows a simple shape sequence that is suitable for use as a status indicator for a two-state status point. (In this example, the first shape represents a "bad value" and is only displayed if the server cannot determine the point's value.)



The following table shows the result when this shape sequence is used in a display to reflect the state of a status point.

State of status point:	The result in the display:
0	
1	

State of status point:	The result in the display:
Unknown (“bad value”)	

The number of shapes must be greater than or equal to the number of parameter states. (Plus an extra shape if you want a “bad value” shape.) If the shape sequence has more shapes than the parameter has states, the remaining shapes are never used. For example, if a parameter has four states and the shape sequence has nine shapes (including a “bad value” shape), only shapes 2 to 5 shapes are used.

Using a shape sequence with an analog point

When a shape sequence is used with an analog point, the shapes are automatically allocated across the point’s range (excluding the first shape if it represents a “bad value”). For example, if the value ranges from 0 to 1000, and there are 11 shapes (the first representing a “bad value”), then:

- Shape 2 is displayed if the value is between 0 and 100
- Shape 3 is displayed if the value is between 101 and 200
- .
- .
- Shape 11 is displayed if the value is between 901 and 1000

Using a shape sequence as an animation

The following figure shows a shape sequence that is suitable for use as an animation, when used in conjunction with scripts. (For details about the scripts, see “Creating an animation with a shape sequence” on page 650.)

Each shape, except the first, represents a separate “frame” in the animation—when displayed rapidly in succession they give the impression of a rotating fan. (The first shape is used to represent the stopped fan.)

About custom properties

There are two types of custom properties, shape custom properties and system custom properties.

By using shape custom properties, you can reuse dynamic shapes or shape sequences across multiple displays. You can map the custom property either when you insert the shape or using scripting.

By using system custom properties, you can create a single generic display that can be reused across your facility. This is useful where you have an enterprise model that uses a consistent structure within the asset hierarchy. (For more information about generic displays, see “About generic displays” on page 61.)

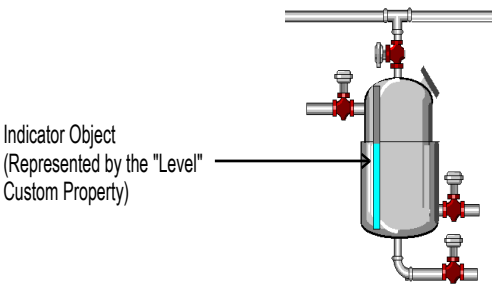
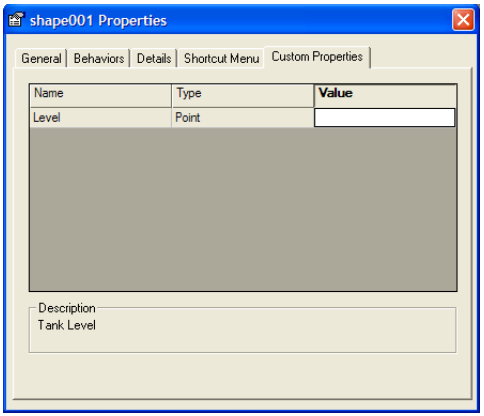
Using shape custom properties to map dynamic objects to points

Each time you insert a dynamic shape into a display, you use shape custom properties to map its dynamic objects to the appropriate points/parameters.

The figure on the left following table shows a typical dynamic shape, in which:

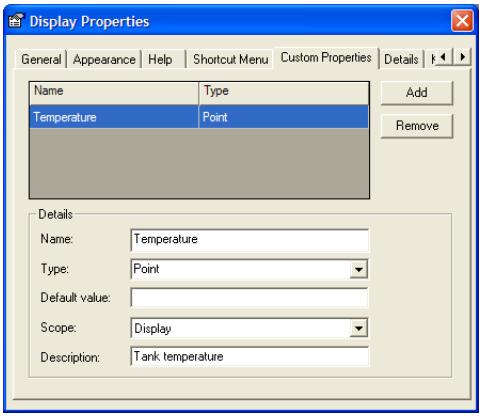
- The indicator is configured to show the PV of one point (which represents the level of liquid in the tank)
- The chart is configured to show the PV and SP of another point (which represent the tank’s actual and desired temperatures)

The figure on the right shows how, when you insert this shape into a display, you map the custom properties to points by typing their names in the **Value** column. (For details, see “Inserting a dynamic shape” on page 97.)

A typical dynamic shape	Points mapped to the custom properties when the shape is inserted into a display
	

Using system custom properties to map objects to points

When you insert an object into a display, you use system custom properties to map the object to the appropriate points/parameters.

A typical object	Points mapped to the custom properties when the object is inserted into a display
Alphanumeric Objects (Represented by the “Level” System Custom Property) 	

About popups and faceplates

A *popup* is a secondary window that appears when a user clicks the object to which it is attached.

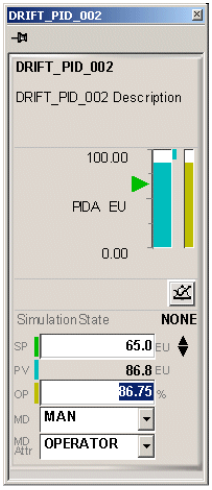
A *faceplate* is a specialized type of popup that shows critical information about the point to which the object is linked.

In most cases, a faceplate is similar to the left-hand portion of the matching point detail display. As with point detail displays, users can control points—for example, change a point's SP—providing they have the required authority.

Notes

- To learn how to create a faceplate, see "Creating a faceplate" on page 50.
- To learn how to attach a popup or faceplate to an object, see "Attaching a popup or faceplate to an object" on page 100.
- Station can simultaneously display up to four faceplates. If a user clicks a fifth object, the oldest faceplate is replaced by the newest one.

Figure 4 Typical faceplate



About display templates

Display templates make it easier to create new displays and to keep them visually consistent. (A template is a display that is stored in the HMIWeb Display Builder's `Templates` folder.)

For example, if you need to create a set of displays that require the same background image and the same set of buttons, you would create a display that contains the image and the buttons, and then save it in the `Templates` folder. Then, when you want to create one of these displays, you simply select the template. (See "Creating a new display" on page 44.)

Notes

- Displays are not linked to the templates. For example, displays do not change if you subsequently change the contents or layout of a template on which they were based.
- You use a template to create other templates. For example, if you wanted to use the same background image on all your displays, you could create a "base" template that contained the image. Then each time you wanted to create a new template, you would select the base template.

Supplied templates

Experion is supplied with the following templates.

Template	Description
Point Detail	A point detail display is used to display point information for all points of a particular type. You need to create point detail displays if you have any specialized point types. For details, see "Creating a point detail display" on page 51. (Experion is supplied with point detail displays for common point types.)

About generic displays

If you have implemented a hierarchical asset model, you can create generic displays that can be used across your plant. For example, if you had a series of holding tanks within your plant, it is likely that the asset structure is repeated. A single display could be created that references this repeated structure and reused to display data for any of the holding tanks. Using the full item name (instead of just the tagname or point ID) in conjunction with system custom properties enables you to create generic displays.

At run time, the value of system custom properties can be stored in either the Station data repository or the display data repository.

If the scope of the system custom property is set to display, when you call up the display in Station, you set the value of the system custom property. The command for calling up a display containing a custom property is *displayname*?*custompropertyname*=*value*, where *displayname* is the name of the display you are calling up, *custompropertyname* is the name of the custom property contained in the display, and *value* is the value you want to set for the custom property.

If the scope of the same system custom property was set to Station, you could create an application script to set the value of the system custom property when Station starts.

For scripting purposes, you specify how system custom properties are accessed; either from the display data repository or the Station data repository. You do this when you set the scope of the system custom property. If you choose the display scope, the property is accessed while the display is called up in Station. If you choose the Station scope, the property can be accessed at any time while Station is running.

For more information about system custom properties, see “About custom properties” on page 57.

Example scenario

You have 3 holding tanks, you want to show the PV of the level of each tank (as an alphanumeric and as a level indicator that is part of a shape) on a separate display. The tanks share a common location in the asset model, `/Assets/Train1/HoldingTanks`. The item names of the tanks are `Tank1`, `Tank2`, `Tank3`. Each tank contains a level indicator point with an item name of `Level`. You want operators to specify which tank they want to view when they call up the display.

Solution

Create a new custom display using HMIWeb Display Builder. Create a system custom property called `HoldingTank` with a scope of `Display`, which will

represent part of the full item name of the tank. The alphanumeric object will be mapped to this system custom property. Create a dynamic shape with a level indicator object and assign a shape custom property called `TankLevel`. The shape custom property is then mapped to the system custom property.

To create the system custom property:

1. Create a new display.
2. Without anything selected, double-click the display.
   The display Properties Window opens.
3. Click the **Custom Properties** tab.
4. Click **Add**.
5. Type **`HoldingTank`** in the name box.
6. Select **Point** as the type.
7. Type **`Tank1`** as the default value.
8. Select **Display** as the scope.
9. Click **Save** and save the display as `HoldingTanks.htm`.

Add an alphanumeric to the display, bind the alphanumeric to `/Assets/Train1/HoldingTanks/<HoldingTank>/Level` and set the parameter to PV.

To bind the alphanumeric:

1. Place an alphanumeric on the custom display and open the Properties Window for the alphanumeric.
2. Click the **Data** tab.
3. Type **`/Assets/Train1/HoldingTanks/<HoldingTank>/Level`** in the Point box.
4. Select **PV** as the parameter.
5. Save the display with the name `HoldingTanks.htm`.

Test the display in Station by calling up the HoldingTanks display showing the PV for `Tank3`.

To call up the HoldingTanks display in Station:

1. In the Command Zone in Station type **`HoldingTanks.htm?HoldingTank=Tank3`**.

The alphanumeric is bound to the point with the full item name of `/Assets/Train1/HoldingTanks/Tank3/Level`.

To create the shape custom property:

1. Open `3dtank04.sha` from the shape library.
2. Without anything selected, double-click the shape.
   The shape Properties Window opens
3. Click the Custom Properties tab.
4. Click **Add**.
5. Type `TankLevel` in the **Name** box.
6. Double-click the **Type** box and select `Point`.
7. Close the Shape Properties.
8. Add an indicator object to the tank.
9. Double-click the indicator object.
   The Properties Window for the indicator object opens.
10. Click the **Data** tab.
11. From the Point list, select `<TankLevel>`.
12. Group the objects.
13. Choose **File > Save As** and the save the shape file as `TankLevel.sha`.

To insert the shape and map the shape custom properties:

1. Open the display with the name `HoldingTanks.htm` that you created earlier.
2. Click on the Toolbox toolbar.
3. Move the pointer to the display and drag it diagonally to mark the rectangle where you want to insert the shape. (The top-left of this rectangle marks the top-left of the inserted shape.)
   When you release the mouse button, the **Insert Shape** dialog box appears.
4. Select `TankLevel.sha` and select the appropriate **Insert into display** option and then click **Open**.
   The shape appears in the display.
5. Double-click the shape.
   The Properties Window opens.
6. Click the **Custom Properties** tab and set the value of the `TankLevel` shape custom property to `/Assets/Train/HoldingTanks/<HoldingTank>/Level`.

7 Close the Properties Window.

8 Save the display.

Test the display again in Station by calling up the `HoldingTanks` display showing the PV for `Tank3`.

To call up the HoldingTanks display in Station:

1 In the Command Zone in Station type **`HoldingTanks.htm?Tank=Tank3`**.

The alphanumeric and the level indicator are bound to the point with the full item name of `/Assets/Train1/HoldintTanks/Tank3/Level`.

About archive format

If you need to keep track of changes to displays—for example, you use a version management tool such as Microsoft Visual Source Safe—you should save them in archive format.

When you save a display/shape in archive format, it is saved as a single file, which is suitable for use with version management tools. (If you save a display in standard format, an associated folder is also created that contains various support files and graphics. The folder has the same name as the display, but with a "`_files`" extension.)

Archived files have a "`hda`" (display) or "`hsa`" (shape) extension.

Notes

- Because you cannot call up an archive file in Station, it is a good idea to set up HMIWeb Display Builder so that displays are simultaneously saved in both standard and archive format. To do this, choose **Tools** > **Options** and select **Save unarchived copy of file with archive files**.
- If you need to archive/unarchive numerous displays or shapes, see "Using the HMIWeb Display Builder Assistant" on page 271.
- Archive format is useful if you want to email a display to someone.

To save a display/shape in archive format:

1. Choose **File** > **Save As** (or **Save** if saving for the first time).
2. Select `Display Archives` from **Save as type**.
3. Specify the filename and click **Save**.

Drawing techniques

3

This chapter describes standard drawing techniques, such as creating, grouping and resizing objects.

Many of the drawing techniques are similar to those used in other Windows-based drawing and paint applications.

Creating objects

You use the Object-creation buttons on the Toolbox Toolbar to create objects.

The way in which you use a tool depends on the type of object it creates. The following procedures describes typical object-creation techniques.

To create a regularly shaped object such as a rectangle or pushbutton:

1. Click the appropriate object-creation button.
2. Drag the pointer diagonally over the display, and release the mouse button when the dotted “outline rectangle” is the correct size and shape.

To create a square/round object, hold down SHIFT while dragging.

To create a straight line:

1. Click
2. Drag to mark the length and direction of the line.
3. Release the button when the line has the correct length and direction.

To draw a line that it is either horizontal, vertical or 45 degrees, hold down SHIFT while dragging. (This also applies when creating segments of a polygon or polyline.)

To create a polygon:

1. Click
2. Click to mark each node, except for the last.
3. Double-click to mark the last node, which also closes the polygon.

To create a polyline:

1. Click
2. Click to mark each node, except for the last.
3. Double-click to mark the last node.

To create a textbox:

1. Click
2. Drag the pointer diagonally over the display, and release the mouse button when the dotted “outline rectangle” is the correct size and shape.

   When you release the mouse button “Text” appears in the box.

3 Delete “Text” and type your text.

You can start a new paragraph by pressing **ENTER**.

To create a hyperlink:

1 Click

2 Drag the pointer diagonally over the display, and release the mouse button when the dotted “outline rectangle” is the correct size and shape.

When you release the mouse button “Text” appears in the box.

3 Delete “Text” and type a suitable the name for the link.

(By default, hyperlinks are blue and underlined, but you can change this if appropriate.)

4 While the object is still selected:

a. Open the Properties window. (If it is not visible, click on the View Toolbar.)

b. Click the **Details** tab.

c. Specify the link’s target, either a Web page or another display. See “Hyperlink” on page 185.

Creating a Bezier curve

You use the **Bezier** tool——to create *Bezier curve* (a type of line), such as the one shown in the following figure.

Notes

- A Bezier curve has curve control handles in addition to start and end nodes—see "Curve control handles" on page 71.
- To change the shape of a bezier curve after creating it, see "Using the Node Editor" on page 88.

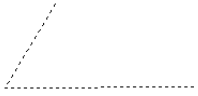
To create a Bezier curve:

1 Click on the Toolbox toolbar.

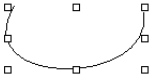
2 Click in the display to mark the first node (start of the curve).

3 Click to mark the first curve control handle. This handle controls the direction at which the curve leaves the first node.

4 Click to mark the second curve control handle. This handle controls the direction at which the curve approaches the second node (end of the curve).



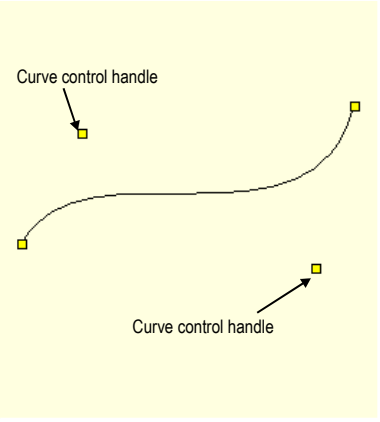
5 Click to mark the second node (end of the curve).

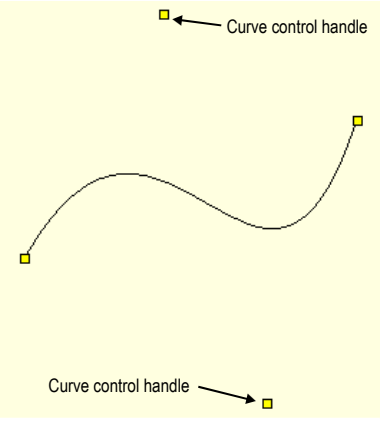


Curve control handles

A Bezier curve has *curve control handles* that control the path the curve takes between nodes. The following curves, which have the same start and end points, show how a curve control handle determines:

- The angle at which the curve leaves/reaches the associated node—initially, it goes in the direction of the curve control handle before going towards the other node
- The amount the curve deviates from a direct path to the other node—the further the curve control handle is away from its node, the further the curve deviates before going towards the other node





Creating a new object from overlapping objects

You can create a new vector graphic object from two or more overlapping vector graphic objects. The following table shows typical results—note, however, that the results depend on the object types and the way they overlap.

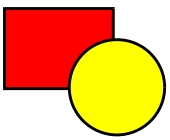
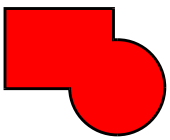
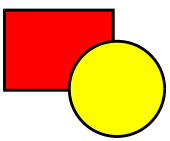
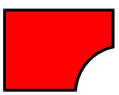
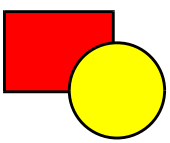
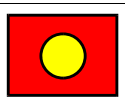
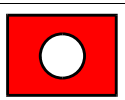
Notes

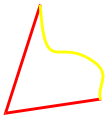
- To learn about vector graphic objects, see “About vector graphic objects” on page 222.
- The original objects are deleted.
- The new object inherits the properties—such as the name, color and line thickness—of the reference object (the first object you select). It also inherits the scripts and dynamic data binding of the reference object.
- The and buttons are used to combine objects to improve the display’s performance—see Combining objects to improve display performance.

To create a new object:

1 Select the overlapping objects.

2 Click the appropriate button on the Transform toolbar.

If you start with this (the darker object is selected first):	And click this:	You get this:
		
		
		
		

If you start with this (the darker object is selected first):	And click this:	You get this:
	*	

* You can only use this button with “open” objects, such as lines and polylines. It is generally used to create a “closed” object from two open objects. (You can then, for example, apply a fill color to the new closed object.)

Selecting objects

Notes

- You can also select objects via the Object Explorer.
- If you want to select all objects in the display, choose **Edit** > **Select All**.

To select one or more objects:

1 Click on the Toolbox toolbar.

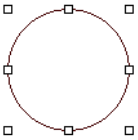
2 Select the object(s) as appropriate.

To select:	Do this:
One object	Click it.
Several objects	Hold down SHIFT and click each object.
Several objects, which are generally close together	Drag the pointer diagonally across the objects. This selects any object that falls entirely within the "selection rectangle" which appears while dragging.
Several objects, some of which are close together	Hold down SHIFT and click each object you want to select. Then, while still holding down SHIFT, drag the pointer diagonally across the other objects—this selects any object that falls entirely within the "selection rectangle" which appears while dragging. You can also reverse the above steps—drag first and then select individual objects.
One object from a stack of objects	Hold down CTRL and click repeatedly until the object you want is selected. Each time you click, the next object in the stack (starting from the top) is selected. Do not click too quickly—if you do, your action will be interpreted as a double-click.

Selection handles

When you select an object, a set of eight selection handles appears around the object's boundary. You use these handles when resizing the object.

Figure 5 A selected object



Reference object

If you select more than one object, the first object you select is called the *reference* object. Some drawing techniques make use of a reference object—for example, you can resize objects to make them the same height/width as the reference object. (The handles of the reference object are different from the handles of the other objects.)

Deselecting objects

To deselect:	Do this:
All objects	Click a blank part of the display. (If an object, such as a "background bitmap", covers the entire display, choose **Edit** > **Deselect All**.)
One or more objects from a group of selected objects	Hold down SHIFT and click each object you want to deselect.

Deleting objects

To delete one or more objects:

1. Select the object(s).
2. Press DELETE.

Resizing objects

To resize one or more objects:

1 Select the object(s). Their selection handles appear.

2 Drag one of the selection handles as required. The objects are resized as you drag.

If you want to retain the object's proportions, hold down SHIFT and drag a corner handle.

To resize objects so that they are the same size as a reference object:

1 Select the reference object.

2 Select the objects you want to resize.

3 Click the appropriate button on the Arrange toolbar.

To make the objects the same:	Click this:
Width as the reference object	
Height as the reference object	
Size (width and height) as the reference object	

Duplicating objects

Notes

- The duplicated object(s) is identical to the original except for the name and position—the duplicated object is offset with respect to the original.
- The object's scripts are also duplicated, but any reference to the original object is changed to the duplicated object. For example, if the original script changes the original object's color, the duplicated script will make the same change to the duplicated object's color. However, you should carefully check all duplicated scripts to check whether you need to make any further changes.

To make duplicate copies of one or more objects:

1 Select the object(s).

2 Click on the Standard toolbar.

Locking objects

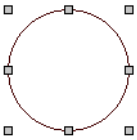
Locking objects in place prevents them from being accidently moved or modified with the mouse. (You can still move and resize a locked object by editing its size and position properties in the Properties Window.)

To lock one or more objects:

1 Select the object(s).

2 Click on the Arrange toolbar. Note that the selection handles turn grey to indicate that the object is locked.

Figure 6 The selection handles of a locked object



To unlock a locked object, select it and then click a second time.

Aligning objects

To align objects with respect to a reference object (the reference object does not move):

1 Select the reference object.

2 Select the objects you want to align.

3 Click the appropriate button on the Arrange toolbar.

To move the objects:		Click this:
Horizontally so that:	Their left edges are aligned to the left edge of the reference object	
	Their centers are aligned to the center of the reference object	
	Their right edges are aligned to the right edge of the reference object	
Vertically so that:	Their tops are aligned at the top edge of the reference object	
	Their centers are aligned to the center of the reference object	
	Their bottoms are aligned at the bottom edge of the reference object	

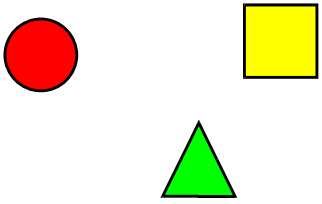
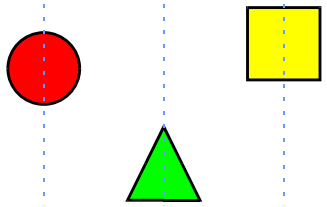
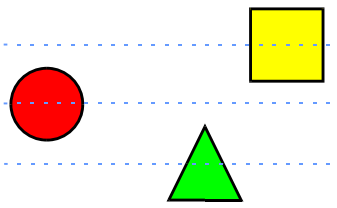
Distributing objects

Distributing objects means moving them so that their centers are equally spaced horizontally or vertically. When you distribute objects, the outer two objects don't move—for example, if you select three objects, the inner object moves left/right (or up/down) so that is centered between the two outer objects.

To distribute three or more objects:

1 Select the objects.

2 Click the appropriate button on the Arrange toolbar.

If you start with this:	And want this:	Click this:
		
		

Grouping objects

You can create a complex object by creating a number of simple objects, arranging them in an appropriate manner and then grouping them. If appropriate, you can "nest" grouped objects to create even more complex objects.

Notes

- When you group a set of objects, you create a *group object*, which is like any other object—that is, it appears in the Object Explorer, has its own properties and can have its own scripts.
- If you ungroup a group, you delete the group object, including any scripts attached to it.

To group two or more objects:

1. Select the objects.
2. Click on the Arrange toolbar.

   If the Object Explorer is visible, note that a new group object is created.

To ungroup a group object:

1. Select the group object and then click .

Adding objects to (or removing them from) an existing group

You can add objects to (or remove them from) an existing group. The advantage of adding/removing objects is that the group (and any scripts attached to it) is not deleted.

To add one or more objects to an existing group:

1. Select the group and the object(s) you want to add.
2. Choose **Arrange > Grouping > Add to Group**.

To remove one or more objects from a group:

1. Using Object Explorer, select the object(s) you want to remove from the group.
2. Choose **Arrange > Grouping > Remove from Group**.

Combining objects to improve display performance

You can improve the performance of a display by combining static vector graphic objects—this converts them into a single metafile, which is easier to render.

Notes

- To learn about vector graphic objects, see “About vector graphic objects” on page 222.
- Combining objects disables any scripts and database links they may have.

To combine vector graphic objects:

1. Select the objects.
2. Click on the Arrange toolbar to group the objects. (You must group the objects before combining them.)
3. Click on the Transform toolbar.

   If the Object Explorer is visible, note how the group is converted into a single object—that is, it changes from this

to this

.

To uncombine a previously combined object:

1. Select the object and then click .

   The object reverts to a group object. If necessary, you can then click on the Arrange Toolbar to ungroup the objects within the group.

Flipping objects

To flip one or more objects:

1 Select the object(s).

2 Click the appropriate button on the Arrange toolbar.

To flip the object(s):	Click this:
Horizontally (as in a mirror image)	
Vertically (turn it upside down)	

Rotating objects

You can rotate textboxes, graphics and vector graphic objects (to learn about vector graphic objects, see “About vector graphic objects” on page 222).

Notes

- If you rotate a group or a shape that contains objects that cannot be rotated, those objects will not rotate.

To rotate one or more objects 90 degrees:

1 Select the object(s).

2 Click the appropriate button on the Arrange toolbar.

To rotate the object(s):	Click this:
90 degrees clockwise	
90 degrees anti-clockwise	

To rotate an object through any angle:

1 Select the object(s).

2 Click on the Toolbox toolbar.

Four rotation handles appear.

3 Drag one of the rotation handles clockwise or anticlockwise. As you drag the handle, the object rotates about its center.

Defining an object's behaviors

A *behavior* is an add-on to an object that changes its appearance or operation. For example, the Popup behavior causes a popup to appear when a user clicks the object.

Notes

- For a description of the behaviors supplied with HMIWeb Display Builder, see "Standard behaviors" on page 165. Note, however, that other behaviors may also be supplied for specialized tasks.
- Extra tabs may appear when you select a behavior—you use them to define the behavior's properties.

To define a behavior:

1 Select the object and click the **Behaviors** tab of the Properties Window.

2 Select the behavior you want to apply.

3 If any new tabs appear, select them and define the behavior's properties.

Changing the stacking order

When objects overlap, you may need to change the order in which they are *stacked* on top of each other—unless an object is transparent, it will obscure objects stacked underneath it. (Objects are initially stacked in the order in which you create them—that is, each new object is added to the top of the stack.)

To move one or more objects up or down the stack:

1 Select the objects.

2 Click the appropriate button on the Arrange toolbar.

To move the object(s):	Click this:
To the front (top) of the stack	
To the back (bottom) of the stack	
One layer up (towards the top of) the stack	
One layer down (towards the bottom of) the stack	

Using the Node Editor

You use the **Node Editor**——to change an object's shape by moving or adding *nodes*. (The nodes are the object's apexes or reference points).

Notes

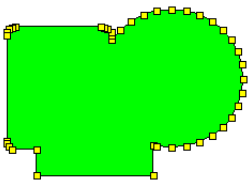
- Some types of object, such as rectangles and ovals, must be first converted to a *path* (line) before you can select the **Node Editor**. To convert an object to a path, select it and then click on the Transform toolbar.

Editing an object's nodes

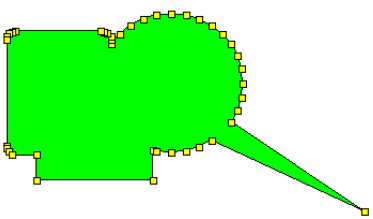
To edit an object's nodes:

1 Select the object.

2 Click on the Toolbox toolbar. A small square marks each of the object's nodes.



3 Move the mouse pointer over a node and drag the node as required.



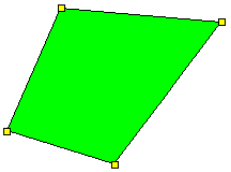
Adding a node

You can also change an object's shape by adding a node between two existing nodes.

To add a node:

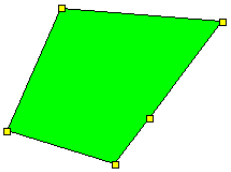
1 Select the object.

2 Click on the Toolbox toolbar.

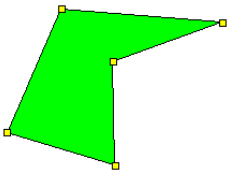


3 Move the mouse pointer over the line between two existing nodes.

4 Click when the pointer turns into a "cross-hair"—this adds the new node.



5 You can now move this node like any other node.



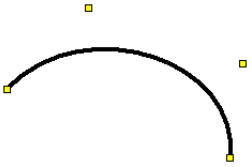
Editing an object's curve control handles

Some types of object, such Bezier curves, have curve control handles that control the path the line takes between nodes. See "Curve control handles" on page 71.

To edit an object's curve control handles:

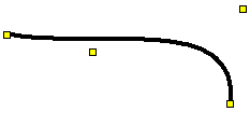
1 Select the object.

2 Click on the Toolbox toolbar.



3 Move the mouse pointer over a curve control handle and drag it to a new position.

4 Release the curve control handle to see the result.



5 If the result is not satisfactory, repeat steps 3 and 4.

Moving objects with the arrow keys

If you want to move an object a small distance—a technique called *nudging*—you may find it easier to use the arrow keys than to use the mouse. Each time you press an arrow key, the selected object(s) move one pixel in the direction of the arrow if the snap-to-grid function is off, or one grid unit if it is on.

If you hold down an arrow key, the selected object(s) move slowly in the direction of the arrow. If you hold down SHIFT and an arrow key, the objects move five times faster.

Inserting a graphic

Inserting digitized photographs, schematic drawings and other suitable graphics (pictures) can significantly enhance your displays.

Notes

- Inserted graphics are copies of the original graphics, which means that the display will not be updated if you subsequently change the original graphics.
- Inserting a large graphic, or many small ones, may affect Station's performance when the display is called up.
- To learn about the graphic formats that HMIWeb Display Builder supports, see "About graphic formats" on page 93.

To insert a graphic:

1 Click on the Toolbox toolbar.

2 Move the pointer to the display and drag it diagonally to mark the rectangle where you want to insert the graphic.

3 When the rectangle is the correct size/shape and you release the mouse button, the **Insert Picture** dialog box appears. (The graphic takes on the size and shape of the rectangle that appears while you are dragging.)

4 Select the graphic and click **Open** to insert it.

5 If necessary, move and resize the graphic.

Inserting a graphic from the clipboard

If you are working simultaneously with HMIWeb Display Builder and another drawing application, you can insert graphics via the clipboard.

To insert a graphic from the clipboard:

1 Create the graphic and then copy it to the clipboard.

2 Switch to HMIWeb Display Builder and indicate where you want to center the graphic by clicking the center point.

3 Click to paste it into the display.

The **Save As** dialog box appears. (Save the graphic with a suitable name.)

The inserted graphic has the same size and shape as in the other drawing application.

4 If necessary, move, resize and group the graphic.

About graphic formats

The ideal format for a particular graphic depends on several factors, such its contents (such as a photograph or schematic drawing), its size and its complexity. In practice, it is a good idea to perform some tests to determine which format best suites a particular need.

There are two basic types of graphic format:

- **Bitmap**. Bitmaps consist of an array of colored pixels. Examples of bitmaps are digitized photographs and screen captures.

  Bitmaps are created in "paint" programs such as Adobe Photoshop and Jasc Paint Shop Pro.

- **Vector graphic formats**. Vector graphics consist of "line-based" objects similar to HMIWeb's inbuilt vector graphic objects—see "About vector graphic objects" on page 222. Examples of vector graphics are schematic drawings and floor plans.

  Vector graphics are created in "drawing" programs such as Autodesk AutoCAD, Adobe Illustrator and Microsoft Visio.

The following scenarios illustrate how to choose an appropriate format:

- You have an AutoCAD drawing of a plant layout that you want to use as a background to your display. In this case, you save the graphic in WMF or EMF format.
- You want to include a high-quality photograph in your display. In this case, you save the photograph in JPG format.
- You want to have overlapping objects, but don't want an object on top to totally obscure the one below. In this case, you save the graphic in PNG format.

The following table lists the graphic formats supported by HMIWeb Display Builder, and summarizes some of their major distinguishing features.

Format	Features
Bitmap	
GIF (`*.gif`)	Only supports 256 colors. Very small file size. Only supports on/off transparency.
Portable Network Graphic (`*.png`)	Supports many colors. The file size depends on the complexity of the image and number of colors used. Supports 256 levels of transparency.
Windows Bitmap (`*.bmp`)	

Format	Features
JPEG (`*.jpg`)	Optimized for photographs and other continuous tone images. Note that JPEG uses “lossy” compression which loses some of the original image data. This means that you should keep the original image (at least until the display has been thoroughly tested) in case you need to make any changes.
Vector graphic	
Metafile (`*.wmf`)	Suitable for schematic drawings and floor plans.
Enhanced Metafile (`*.emf`)	An enhanced version of WMF. Suitable for schematic drawings and floor plans.

Inserting text from another application

You can use the clipboard to insert text from other applications. For example, if you want to include a large block of text in your display, you may find it easier to compose it in a text editor or word processor.

Notes

- Formatting is ignored when the text is pasted from the clipboard. (The Paste function ignores everything except the text.)
- If the text includes HTML/XML tags, the effect may be undefined.

To insert text into a display:

1. Copy the text to the clipboard from the other application.
2. Switch to HMIWeb Display Builder and indicate where you want to center the text by clicking the center point.
3. Click to paste it into the display.
4. If necessary, move, resize and group the text.

If the text block is large, it is easier to use the Properties Window to resize the block.

Inserting a shape sequence

Notes

- To learn about shape sequences, see “About shape sequences” on page 55.
- When you insert a shape sequence, you can either:
  - **Embed it**. This is the recommended option because it optimizes display performance. (This pastes a copy of the shape into the display.)

    You can make this the default option by choosing **Tools** > **Options**, clicking the **Shapes** tab and selecting the **Contents of shape file** option.
  - **Link to it**. (The shape remains in a separate file.)
- You cannot insert a linked shape sequence unless you have registered the folder that contains it. See “Setting HMIWeb Display Builder’s options” on page 266.
- You can also insert a shape sequence by dragging the shape sequence from the Shape Gallery.

To insert a shape sequence:

1. Click on the Toolbox toolbar.
2. Move the pointer to the display and drag it diagonally to mark the rectangle where you want to insert the shape. (The top-left of this rectangle marks the top-left of the inserted shape.)

   When you release the mouse button, the **Insert Shape** dialog box appears.
3. Select the shape sequence you want to insert, select the appropriate **Insert into display** option and then click **Open**.

   The shape appears in the display.
4. If necessary, move and resize the shape.
5. Open the Properties Window, click the **Behaviors** tab and select **Shape Sequence Animation**.

   The Data and Animation tabs are added to the Properties Window.
6. Click the **Data** tab and enter the database link details. For example, if you are linking to a particular point parameter, select `Point/Parameter` from **Type of database link**, and then enter the appropriate point and parameter details.
7. Click the **Animation** tab and enter the number of shapes you want to use in **Number of shapes**. (For example, if the parameter has only four states but the shape sequence contains eight shapes, type **4** to use the first four shapes.)

Inserting a dynamic shape

Notes

- To learn about dynamic shapes, see “About dynamic shapes” on page 54.
- When you insert a dynamic shape, you can either:
  - **Embed it**. This is the recommended option because it optimizes display performance. (This pastes a copy of the shape into the display.)

    You can make this the default option by choosing **Tools** > **Options**, clicking the **Shapes** tab and selecting the **Contents of shape file** option.
  - **Link to it**. (The shape remains in a separate file.)
- You cannot insert a linked dynamic shape unless you have registered the folder that contains it. See “Setting HMIWeb Display Builder’s options” on page 266.
- You can also insert a dynamic shape by dragging the dynamic shape from the Shape Gallery.

To insert a dynamic shape:

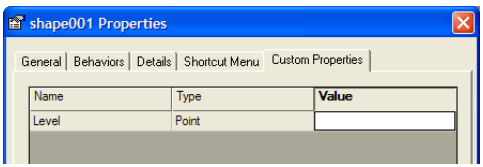
1. Click on the Toolbox toolbar.
2. Move the pointer to the display and drag it diagonally to mark the rectangle where you want to insert the shape. (The top-left of this rectangle marks the top-left of the inserted shape.)

   When you release the mouse button, the **Insert Shape** dialog box appears.
3. Select the dynamic shape you want to insert, select the appropriate **Insert into display** option and then click **Open**.

   The shape appears in the display.
4. If necessary, move and resize the shape.
5. Open the Properties Window, click the **Custom Properties** tab and, for each custom property, specify an appropriate value.

   Depending upon the type of the custom property, appropriate values include point or parameter names and references to other custom properties.

   The custom properties, and what they represent, are specific to each shape. In the following example, you must specify a point ID for **Level**.



Embedding a linked shape

If a shape is linked to a display, you can *embed* it. (Embedding pastes a copy of the shape into the display.)

Notes

- Embedding is recommended because it optimizes display performance.
- Unlike a linked shape, an embedded shape is not automatically upgraded if you subsequently change the shape. If you want to upgrade an embedded shape, see "Upgrading an embedded shape" on page 99.

To embed one or more linked shapes:

1. Select the linked shape(s).

   If you want to select a lot of shapes, it is easier to select them using the Object Explorer.
2. Choose **Tools** > **Embed Shapes**.
3. Click **OK**.

Upgrading an embedded shape

If you embed a shape into a display and subsequently change the shape, you may want to upgrade the embedded shape in the display.

Notes

- The following procedure only works if the shape file is in a registered folder. See “Setting HMIWeb Display Builder’s options” on page 266.
- You can use the HMIWeb Display Builder Assistant to upgrade embedded shapes in every display within a specified folder. See “Using the HMIWeb Display Builder Assistant” on page 271.

To upgrade one or more embedded shapes:

1 Select the embedded shape(s).

If you want to select a lot of shapes, it is easier to select them using the Object Explorer.

2 Choose **Tools** > **Upgrade Embedded Shapes**.

3 Click **Upgrade**.

Attaching a popup or faceplate to an object

Notes

- To learn about popups and faceplates, see “About popups and faceplates” on page 59.
- In order to preview a popup/faceplate in a display, you must register the folder in which you store the popup in the **Shapes** tab of the **Options** dialog box. See “Setting HMIWeb Display Builder’s options” on page 266.
- Because operators may not know that a popup/faceplate is attached to an object (other than moving the mouse pointer over the object), you should consider using a visual cue. For example, if a popup/faceplate is attached to an alphanumeric, you could set its border color to red (and set the border color of the other alphanumerics to black).

To attach a popup/faceplate to an object:

1. Select the object.
2. Click the **Behaviors** tab of the Properties Window and select either **Popup** or **Faceplate**, as appropriate.
3. If you selected **Popup**, click the **Popup** tab and specify the popup’s filename and initial position.

   (If you selected **Faceplate**, no extra configuration is required because the appropriate faceplate for the type of point to which the object is linked is automatically attached.)

Inserting an ActiveX control

Attention

Because of the extremely varied nature of ActiveX controls, you should thoroughly test the operation of displays that include ActiveX controls before releasing them to users.

Notes

- The effects of inserting an ActiveX control vary widely, according to the nature of the control. Some controls may, for example, initiate downloading and installation of control-related components.
- If any security-related warning appears, check that it is appropriate to the control you are inserting before proceeding. If you are unsure, you should refer to the relevant Microsoft documentation.
- Some controls add specialized properties to the Properties Window, which you must define. (See the documentation/help supplied with the control for details.)

To insert an ActiveX control:

1 Click the arrow to the right of on the Toolbox toolbar to select the control.

2 Drag diagonally across the display to mark size and position of the “window” for the control.

3 Open the Properties Window and configure the control’s properties as appropriate.

Inserting an ActiveX document

You can insert an ActiveX document into a “window” that is embedded into a display. (Scrollbars appear if the document is larger than the window).

ActiveX documents include Microsoft Word documents, and Microsoft Excel spreadsheets.

To insert an ActiveX document:

1. Click on the Toolbox toolbar and then drag diagonally across the display to mark size and position of the window for the document.
2. Open the Properties Window and click the **Details** tab.
3. Type the name of the document in **File Name** (or click **Browse** and search for the document).

Defining an object's color

You use the and buttons on the Drawing toolbar to define an object's line and fill colors.

For example, to change the fill color of one or more objects:

1 Select the object(s) whose line color you want to change.

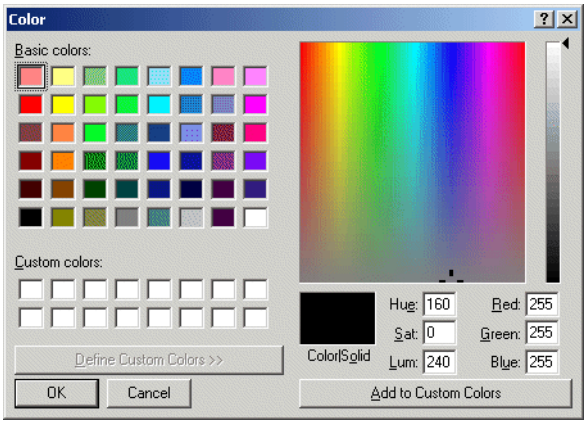
2 If you want to:

- Apply the default color (as shown on the button), click .
- Apply a standard color, click the small arrow to the right of and click the color.
- Apply a non-standard color, see "Selecting a non-standard color" on page 104.
- Remove the existing color, so that the line becomes invisible (transparent), click the arrow to the right of and then click **No Line**.
- Apply a gradient fill (where two colors blend into each other), see "Applying a gradient fill" on page 105.

Selecting a non-standard color

For example, to select a custom fill color:

1 Click the small arrow to the right of and click **More Fill Colors**.



2 Click the color you want in the “color spectrum” box and then adjust the slider on the “luminance box” (the thin bar on the right).

Alternatively, type the HSL or RGB values in the appropriate boxes.

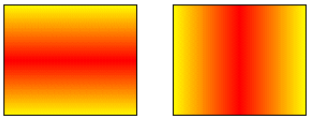
3 Click **Add to Custom Colors** so that the color appears in the next available **Custom colors** box.

4 Click the custom color you have just created and then click **OK**.

Applying a gradient fill

With a gradient fill, one color is applied to the center of the object and another color is applied to the edges (top/bottom or left/right). In between, the color gradually changes between the two colors.

Figure 7 Typical gradient fills—horizontal (left) and vertical (right)



Notes

- You can only apply a gradient fill to some types of object, such as rectangles, ovals and wedges.

To apply a gradient fill to one or more objects:

1. Select the objects.
2. Click the small arrow to the right of and click **Fill Effects**.
3. Select **Horizontal Gradient** or **Vertical Gradient** as appropriate.
4. Select the appropriate **Fill Color** (center color) and **Gradient Color** (edge color).
5. Select the appropriate **Focus**. (The focus defines the relative position between the edges, where the fill color is "pure". 50% represents the center.)

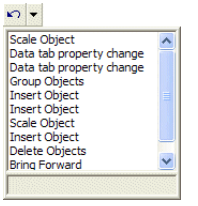
   The Preview box gives an idea of what the gradient will look like.
6. Click **OK**.

Undoing your work

You use the and buttons on the Standard toolbar to undo changes you have made to a display—in effect the buttons work in a similar manner to the Forward and Reverse buttons on a video recorder.

Each time you click , you go undo one change in the sequence of changes. If you go back too many steps, to retrace your steps.

If you click the arrow to the right of a button, the list of actions appears (the most recent action is at the top). You can simultaneously undo/redo several actions by moving the pointer down to the oldest action you want to undo and then clicking.



The maximum number of changes you can undo is specified in the **General** tab of the **Options** dialog box—see “Setting HMIWeb Display Builder’s options” on page 266.

Using advanced display features

4

This chapter describes how to use advanced display features, such as style sheets and shortcut menus.

About style sheets

A *style sheet* makes it much easier to create visually consistent displays.

A style sheet contains a set of *styles* (definitions) that control the presentation-related properties of objects, such as color, line thickness and font. A style sheet can also include a style that controls the display's background properties such as its background color.

For example, if you wanted some alphanumerics to have a gray background and use 10 point Courier, you would first add a style to your style sheet that defined those properties; you would then select that style for the relevant alphanumerics.

Another major advantage of using a style sheet is that if you update it, the changes are automatically applied to the associated displays when they are next called up or refreshed.

Notes

- To learn how to use a style sheet, see "Attaching a style sheet to a display" on page 109.
- To learn how to create a style sheet, see "Creating a style sheet" on page 111.
- Assigning a style to an object or display disables the corresponding properties in the Properties Window. For example, if the **Fill Color** of an object is currently set to red and you apply a style that defines it as blue, the fill color will change to blue.
- Scripts can override values defined in a style by using the style object. For example, the following code changes the fill color of an object, rect001, to green even if it is defined as blue in the style.

  ```
  rect001.style.fillcolor = green
  ```

- You can also use styles to set custom properties and to control the position of objects. See "Special uses for user-defined style attributes" on page 121.

Attaching a style sheet to a display

The following procedure describes how to attach a style sheet to a display and how to attach its styles to objects in the display.

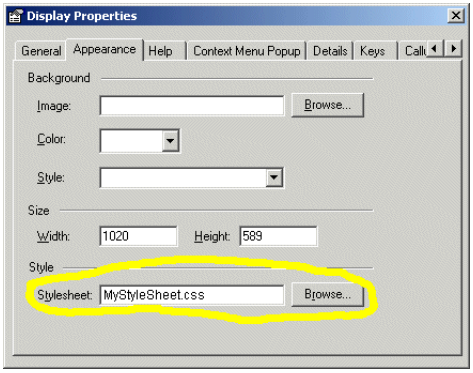
Notes

- In general, you should store style sheets in the same folder as the displays that use them.
- If you specify a style sheet using a relative path and the display has not been saved, the styles will not be available until the display is saved, closed and reopened.

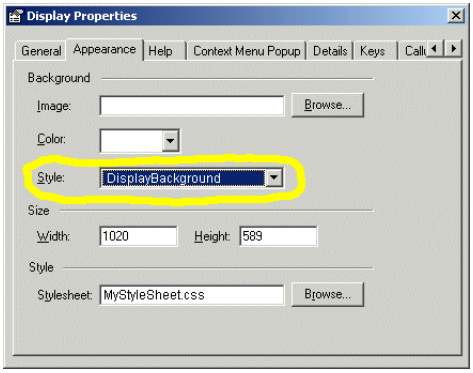
To attach a style sheet:

1 Deselect all objects, and on the **Appearance** tab of the Properties Window, type the name of the style sheet that you want to use. (This assumes that you have the recommended setup, in which the style sheet is in the same folder as the display.)

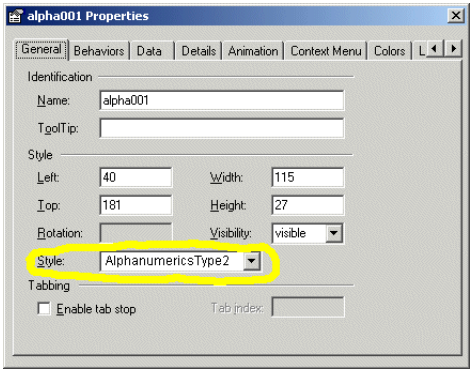
Alternatively, click **Browse** and browse to the style sheet. (Note that this adds the full path to the style sheet. However, you can then edit the path as required—for example to leave only the relative path details.)



2 Select the appropriate background style for the display (if an appropriate style exists).



3 Select one or more objects and, on the **General** tab, select the appropriate style.



4 Repeat step 3 as required for the remaining objects.

Creating a style sheet

You create a style sheet using a text editor such as Notepad.

It is a good idea to base a style sheet on an existing one, such as `sample.css`, which is in `Program Files\Honeywell\Experion PKS\Client\HMIWeb Display Builder\Examples`.

Notes

- You can edit the style sheet attached to the current display by clicking on the View toolbar. (However, any changes you make will not take effect in the display until you close and re-open it.)

Structure of a style

The following code shows the structure of a typical style.

```
.AlphanumericType1
{
  /* Style for Type 1 alphanumerics */
  hw-element-class: hsc.alpha.1;
  hw-text-family: Courier New;
  hw-text-size: 10pt;
  hw-fill-color: #E0E0E0;
  hw-text-color: #000000;
  .
  .
}
```

This style contains the following:

- The style's name, preceded by a period (`.AlphanumericType1`).
- Delimiters ({ and }) that mark the start/end of the definition.
- A comment (`/* Style for Type 1 alphanumerics */`)
- A definition for each *style attribute* that you want to include in the style. For a description of the predefined style attributes, see "Predefined style attributes" on page 114.

  You can also create your own (user-defined) style attributes. See "Special uses for user-defined style attributes" on page 121.

Syntax of a style attribute definition

Each style attribute definition uses the following syntax:

`AttributeName` : *`value`* ;

For a description of the predefined attributes, see "Predefined style attributes" on page 114.

Referencing style sheets used by shape files

If you use a separate style sheet for shape files, you must include a reference to it in the display's style sheet (generally, in the first line of the style sheet).

The syntax of the reference is as follows:

```
@import url("shapestylesheetname.css");
```

Predefined style attributes

The following table describes the predefined style attributes applicable to displays and their objects. In general, each predefined style attribute has an equivalent Properties Window property, as well as an object scripting property. For example, the “hw-fill-color” attribute is equivalent to the “Fill Color” property in the Properties Window and to the “fillColor” scripting property.

For the scripting equivalents, see “Scripting equivalents of the predefined style attributes” on page 119.

Attribute	Description	Applicable to
hw-alpha-format-numeric	Numeric display format for alphanumerics. The value of this attribute follows a subset of the standard C/C++ string formatting rules.*	Alphanumeric
hw-animation-alarm-high-color	Alarm high color. See “Defining colors” on page 118.	Alphanumeric, combobox, vector graphic, indicator
hw-animation-alarm-low-color	Alarm low color.	Alphanumeric, combobox, vector graphic, indicator
hw-animation-alarm-urgent-color	Alarm urgent color.	Alphanumeric, combobox, vector graphic, indicator
hw-animation-breakpointX-class	Stylesheet breakpoint animation class (from 0 to 8). See “Breakpoints” on page 122	Alphanumeric, combobox, vector graphic, indicator
hw-animation-breakpointX-color	Breakpoint animation colors (from 0 to 8).	Alphanumeric, combobox, vector graphic, indicator
hw-animation-offscan-color	Offscan color.	Alphanumeric, combobox, vector graphic, indicator
hw-cursor	The type of mouse pointer used when it is over an object. See “hw-cursor” on page 118.	Page background, alphanumeric, button, checkbox, combobox, vector graphic, indicator, picture, text
hw-element-class	Defines the object types to which a style applies. See “hw-element-class” on page 117.	All display elements
hw-fill-color	Fill color.	Page background, alphanumeric, button, checkbox, combobox, vector graphic, indicator, text

Attribute	Description	Applicable to
hw-fill-effect	Fill effect: none blink	Alphanumeric, button, checkbox, combobox, vector graphic, indicator, text
hw-fill-gradient-color	Fill gradient color.	Vector graphic
hw-fill-gradient-focus	Fill gradient focus, a relative position specified as an integer between 0 and 100.	Vector graphic
hw-fill-gradient-style	Fill gradient style: vertical horizontal	Vector graphic
hw-fill-level-color	Level fill color.	Vector graphic, indicator
hw-fill-style	Fill style: solid transparent gradient	Alphanumeric, button, checkbox, combobox, vector graphic, indicator, text
hw-image-src	The URL or path/name of the image. See “Specifying URLs and file paths” on page 119.	Page background, picture
hw-line-color	Line color.	Alphanumeric, button, checkbox, combobox, vector graphic, indicator, text
hw-line-effect	Line effect: none blink	Alphanumeric, button, checkbox, combobox, vector graphic, indicator, text
hw-line-end-arrow	Line end arrow type: none open closed oval	Vector graphic
hw-line-start-arrow	Line start arrow type: none open closed oval	Vector graphic

Attribute	Description	Applicable to
hw-line-style	Line style. The following are applicable to all the object types listed opposite: transparent solid The following are only applicable to applicable to alphanumeric, button, checkbox, combobox, indicator: inset outset ridge groove dotted dashed double The following are only applicable to applicable to vector and text: shortdash dash longdash dot dashdot dashdotdot	Alphanumeric, button, checkbox, combobox, vector graphic, indicator, text
hw-line-width	Line width in pixels (an integer).	Alphanumeric, button, checkbox, combobox, vector graphic, indicator, text
hw-rotation	Rotation of object, relative to its parent. An integer between 0 and 359.	Vector graphic, shape, group, text
hw-shape-src	The URL or path/name of the shape. See “Specifying URLs and file paths” on page 119.	Shape
hw-text-align	Text alignment: left center right justify	Alphanumeric, button, checkbox, combobox, text
hw-text-color	Text color.	Alphanumeric, button, checkbox, comboxbox, text

Attribute	Description	Applicable to
hw-text-effect	Text effect: none strikeout underline italic bold blink	Alphanumeric, button, checkbox, combobox, text
hw-text-family	Text family name.	Alphanumeric, button, checkbox, combobox, text
hw-text-size	Text size.	Alphanumeric, button, checkbox, combobox, text
hw-visibility	Visibility: inherit visible hidden	Page background, alphanumeric, button, checkbox, combobox, vector graphic, indicator, picture. text

* For more information about C/C++ string formatting rules, see http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vclib/html/_crt_format_specification_fields_.2d_.printf_and_wprintf_functions.asp.

hw-element-class

The hw-element-class attribute restricts the range of object types to which the style applies. (The advantage of using this attribute is that the **Style** list on the Properties Window only shows styles that are applicable to the selected object.)

The following example restricts a style to alphanumerics and indicators.

```
hw-element-class: hsc.alpha.1 hsc.indicator.1;
```

Object type	Description
#page	Display (Applicable for styles that define the display's background properties.)
hvg.base.1	Vector graphic (See "About vector graphic objects" on page 222.)
hvg.group.1	Group
hvg.textbox.1	Textbox
hvg.combined.1	Combined object (For a description of a combined object, see "Combining objects to improve display performance" on page 83.)
hsc.alpha.1	Alphanumeric
hsc.checkbox.1	Checkbox
hsc.button.1	Pushbutton

Object type	Description
`hsc.indicator.1`	Indicator
`hsc.combo.1`	Combobox
`hsc.image.1`	Graphic (such as a photograph)
`hsc.shape.1`	Shape

hw-cursor

The hw-cursor attribute specifies the type of mouse pointer used when it is over an object. The following table describes commonly used values. (See a style reference for other values.)

Value	Description
`auto`	Station (or the browser in which the display is called up) determines which pointer to use, based on the current context.
`crosshair`	Simple crosshair.
`default`	The platform-dependent default pointer. Usually an arrow.
`hand`	Hand.
`text`	Indicates editable text. Usually an I-bar.
`wait`	Indicates that Station (or the browser in which the display is called up) is busy and the user should wait. Usually a watch or hourglass.

Defining colors

You can define a color using:

- A HTML named color. For example:
  `hw-fill-color: deeppink;`
  For details about named colors, see `http://msdn.microsoft.com/library/default.asp?url=/workshop/author/dhtml/reference/colors/colors.asp`.
- A HTML-formatted 24-bit RGB value. The format is: #*rrggbb*, where *rr*, *gg* and *bb* are the hexadecimal values for the red, green and blue components. This example sets the color to blue:
  `hw-fill-color: #0000FF;`

Note that the server-wide alarm colors will, if enabled, override the colors defined in a style. For details, see the *Configuration Guide*.

Specifying URLs and file paths

The following examples show the syntax used by the hw-shape-src and hw-image-src attributes to specify URLs and file paths.

A shape that is in one of Station's display folders:

```
hw-shape-src: url(myshape.sha);
```

A shape that is not in one of Station's display folders:

```
hw-shape-src: url(c:\mydir\myshape.sha);
```

An image that is on a Web site:

```
hw-image-src: url(http://myweb/picture.gif);
```

Scripting equivalents of the predefined style attributes

The following table lists the equivalent object property for each predefined style attribute.

Attribute	Equivalent object property
hw-animation-alarm-high-color	-
hw-animation-alarm-low-color	-
hw-animation-alarm-urgent-color	-
hw-animation-breakpointX-color	-
hw-animation-offscan-color	-
hw-animation-breakpointX-class	-
hw-cursor	style.cursor
hw-element-class	-
hw-fill-color	fillColor
hw-fill-effect	fillColorBlink
hw-fill-gradient-color	GradientFillColor
hw-fill-gradient-focus	GradientFillFocus
hw-fill-gradient-style	GradientFillStyle
hw-fill-level-color	levelFillColor
hw-fill-style	FillStyle
hw-image-src	src
hw-line-color	lineColor
hw-line-effect	lineColorBlink
hw-line-end-arrow	EndArrow
hw-line-start-arrow	StartArrow
hw-line-style	LineStyle

Attribute	Equivalent object property
hw-line-width	LineWidth
hw-rotation	Rotation
hw-shape-src	src
hw-text-align	style.textAlign
hw-text-color	textColor
hw-text-effect	textColorBlink, style.textDecoration, style.fontStyle, style.fontWeight
hw-text-family	style.fontFamily
hw-text-size	style.fontSize
hw-visibility	style.visibility

Special uses for user-defined style attributes

The values of user-defined style attributes can be retrieved by scripts at run time by using the GetStyleClassProperty method on the page or the GetStyleProperty method on an object.

The following example shows how you can use user-defined style attributes to:

- Set the value of a custom property
- Control the position of a shape within a display

This is the user-defined style in the style sheet.

```
.MyShape_Style
{
  /* My user defined style attributes */
  myFormat: %2.3f;
  myTop: 10;
  myLeft: 10;
}
```

This is the script.

```
Sub Page_onpagecomplete
  ' Set a custom property using a value obtained
  ' from the stylesheet linked to the display
  dim strFormat
  strFormat = shape001.GetStyleProperty("myFormat")
  shape001.SetCustomProperty "value", "format", strFormat

  ' Set the position of an object using a value obtained
  ' from the stylesheet linked to the display
  dim strTop, strLeft
  strTop = shape001.GetStyleProperty("myTop")
  strLeft = shape001.GetStyleProperty("myLeft")

  shape001.style.pixelTop = strTop
  shape001.style.pixelLeft = strLeft
End Sub
```

Breakpoints

Instead of being associated with a color, each display element animation breakpoint can be associated with a distinct style sheet class. Attributes defined in the breakpoint style class are applied to the display element when processing determines its state is equivalent to the breakpoint value.

All properties such as “hw-fill-color” and “hw-text-size”, that can be specified in a style sheet for an alphanumeric can be included in a breakpoint style class. This means that as well as animating text color using breakpoints almost any other property of the alphanumeric can also be animated.

The value of each “class” attribute is a text string defining the name of a style sheet breakpoint class. The attributes contained in the breakpoint class will be applied to the display element if its new value places it in the corresponding breakpoint.

The “hw-animation-breakpoint-class” attributes supersede the existing attributes for breakpoint colors. However, the superseded color attributes will still function, but if a new breakpoint class attribute has been designated for a breakpoint value, the new attribute animation will be performed and the color attribute ignored.

Breakpoint animation

When a display element’s value changes, breakpoint animation processing is performed. If a “hw-animation-breakpoint#-class” attribute has been defined for the breakpoint then the class attribute of the display element will be updated by removing any previously applied breakpoint style and the new breakpoint style class will be appended.

This will cause the display element’s attributes to be updated to include the attribute values specified in the breakpoint style class as well as those already specified in the display element’s base style class. Attributes in the breakpoint class will take precedence over the same attributes in the base style class.

Note that the base style class can still be accessed and modified independently of the class property by accessing the display element’s “styleClass” property.

Example style sheet file

Notes

- Display elements can be assigned a a base style class. In this example, VectorAndAlpha_Style is a base style class.
- The the “hw-animation-breakpoint#-class” properties in the base class associate breakpoints with breakpoint style classes and “Brkpnt#” are the breakpoint style classes.

- To make use of this stylesheet within a display page, you should assign the style VectorAndAlpha_Style style class to a display element which has a breakpoint animation turned on in it's property pages.

```
.VectorAndAlpha_Style
{
   /* Style class only applies to HVG and Alpha elements because
   of the "hw-element-class" attribute */

   hw-element-class: hvg.base.1 hsc.alpha.1;

   hw-animation-breakpoint0-class: Brkpnt0;
   hw-animation-breakpoint1-class: Brkpnt1;
   hw-animation-breakpoint2-class: Brkpnt2;
   hw-animation-breakpoint3-class: Brkpnt3;
   hw-animation-breakpoint4-class: Brkpnt0;
   hw-animation-breakpoint5-class: Brkpnt1;
   hw-animation-breakpoint6-class: Brkpnt2;
   hw-animation-breakpoint7-class: Brkpnt3;
   hw-animation-breakpoint8-class: Brkpnt0;
   hw-animation-breakpoint9-class: Brkpnt1;
   hw-animation-breakpoint10-class: Brkpnt2;

   hw-fill-color: LightSalmon;
}

.VectorAndAlpha_StyleB
{
   /* Style class only applies to HVG and Alpha elements because
   of the "hw-element-class" attribute */

   hw-element-class: hvg.base.1 hsc.alpha.1;

   hw-animation-breakpoint0-class: Brkpnt3;
   hw-animation-breakpoint1-class: Brkpnt2;
   hw-animation-breakpoint2-class: Brkpnt1;
   hw-animation-breakpoint3-class: Brkpnt0;
   hw-animation-breakpoint4-class: Brkpnt3;
   hw-animation-breakpoint5-class: Brkpnt2;
   hw-animation-breakpoint6-class: Brkpnt1;
   hw-animation-breakpoint7-class: Brkpnt0;
   hw-animation-breakpoint8-class: Brkpnt3;
   hw-animation-breakpoint9-class: Brkpnt2;
   hw-animation-breakpoint10-class: Brkpnt1;

   hw-fill-color: Turquoise;

}


.Brkpnt0
```

```
{
   hw-element-class: hvg.base.1 hsc.alpha.1;
   hw-text-color: Gold;
   hw-text-size: 8pt;
   hw-fill-color: Blue;
   hw-rotation:0;
}

.Brkpnt1
{
   hw-element-class: hvg.base.1 hsc.alpha.1;
   hw-text-color: Magenta;
   hw-text-size: 12pt;
   hw-fill-color: Black;
   hw-rotation:15;
}

.Brkpnt2
{
   hw-element-class: hvg.base.1 hsc.alpha.1;
   hw-text-color: SteelBlue;
   hw-text-size: 16pt;
   hw-fill-color: Chocolate;
   hw-rotation:30;
}

.Brkpnt3
{
   hw-element-class: hvg.base.1 hsc.alpha.1;
   hw-text-color: #FFC0CB;
   hw-text-size: 20pt;
   hw-fill-color: Green;
   hw-rotation:45;
}
```

About shortcut menus

A *shortcut menu* appears when a user right-clicks an object or the display.

By default, the shortcut menu assigned to an object is determined by the type of point to which it is linked. For example, if you link an object to an analog point, the standard shortcut menu for an analog point will be used.

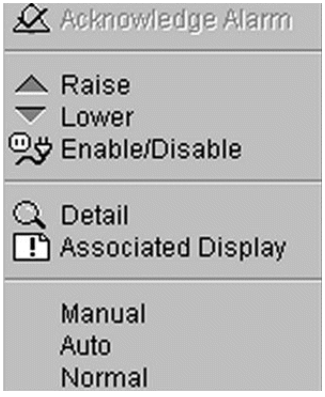
There is also a default display-level shortcut menu that is used if the user right-clicks:

- The display
- An object that is not linked to a point/parameter

Notes

- You use the **Shortcut Menu** tab on the Properties Window to assign a non-default shortcut menu to a display or an object.
- If you want to assign a shortcut menu to a dynamic shape (as a whole), you must assign it to the top-level group within the shape. (By default, each dynamic object uses the menu for the associated point, and any static objects use display's menu.)
- If you want to assign a shortcut menu to a shape sequence, you must assign it to shape in the sequence.
- To learn how to create a custom shortcut menu, see "Creating a custom shortcut menu" on page 126.

Figure 8 Typical shortcut menu (for an analog point)



Creating a custom shortcut menu

In practice, the easiest way of creating a custom shortcut menu (which is a specialized XML file) is to open a standard menu, save it with a new name and edit it as required.

Notes

- To learn about shortcut menus, see "About shortcut menus" on page 125.
- The standard shortcut menus are stored in `Program Files\Honeywell\Experion PKS\Client\System\R`*xxx*, where *xxx* is the release number.
- It is recommended that the filename of a shortcut menu is the same as the matching group faceplate template display (or point detail display), except for the "_sm.xml" suffix. For example, if the filename of the point detail display for a particular type of point is `BoilerType1.htm`, the filename of the shortcut menu must be `BoilerType1_sm.xml`.
- You can use a text or XML editor to create a shortcut menu. However, if you use an XML editor, you can validate it against the DTD (Document Type Definition) for shortcut menus, `shortcutMenu.xsd`.
- If you want to include a special character such as "&" or ">" in a menu item, you must use its *character/entity reference*. For example, if you want a menu item called "Reports & Trends", you would use "&" for the "&". For details, see `http://msdn.microsoft.com/library/default.asp?url=/library/en-us/xmlsdk/htm/xml_concepts_67jn.asp`

The following procedure shows how to create a new menu based on the standard shortcut menu for analog points (`sysdtlana_sm.xml`).

To create a new short menu based on an existing one:

1. Using a text (or XML) editor open `sysdtlana_sm.xml`.
2. Save it with an appropriate filename.
3. Edit the menu as required. See "Structure of a shortcut menu" on page 127.
4. If you want to write scripts for the menu, see "Writing scripts for shortcut menus" on page 130.

Structure of a shortcut menu

The following example shows the basic structure of a shortcut menu.

```
<?xml ... >
<menu ... >
  <script>
    .
    .
  </script>
  <menuitem ... >
  .
  .
  </menuitem>
  .
  .
  <menuitem ... >
  .
  .
  </menuitem>
</menu>
```

Element	Description

`<?xml ... >`
`<menu ...>`
`.`
`.`
`</menu>`
	These define the basic structure of a shortcut menu and, in general, should not be changed.

`<script>`
`.`
`.`
`</script>`
	An optional script not associated with any particular menu item. For an example of how it can be used, see “Scripts not associated with a particular menu item” on page 134.

`<menuitem ...>`
`.`
`.`
`</menuitem>`
	A single menu item.
	Each `<menuitem ... > </menuitem>` pair defines a single menu item. The order in which menu items are defined in the file defines the order in which they appear in the menu.
	For details, see “Structure of a menu item” on page 128.

`<separator/>`	Inserts a horizontal line between menu items.

`<html>`
`.`
`.`
`</html>`
	HTML code which is not parsed by the XML translator, but is rendered directly in the shortcut menu. (When a context menu is invoked, the XML translator first parses the XML code into HTML code, which the browser then uses to render the shortcut menu.)
	For example, you could use HTML-coded labels (instead of <separator/>) to separate menu items.

Structure of a menu item

The following example shows the basic structure of a single menu item.

```
<menuitem id="mnu_det" acceleratorKey="">
  <script type="onclick">
    window.external.application.InvokeMenu2('commandname');
    window.external.CloseShortcutMenu();
  </script>
  <script type="onpage">
    .
    .
  </script>
  <text>menuitemname</text>
  <image>.\Toolbar\iconname</image>
</menuitem>
```

Element	Description

`<script type="onclick">`
`.`
`.`
`</script>`
`<script type="onpage">`
`.`
`.`
`</script>`
	The scripts:
	• The onclick script runs when the user clicks the menu item. It uses the InvokeMenu2 method to invoke the specified command. To find the name of a particular command, see “Finding the name of a Station command” on page 128.
	• The onpage script runs when the page loads. It enables the menu item to be disabled if the command is not currently appropriate, or if the user does not have the required security level.
	If you want to write your own scripts, see “Writing scripts for shortcut menus” on page 130.

`<text>`*menuitemname*`</text>`	The text that appears in the menu.
`<image>.\Toolbar\`*iconname*`</image>`	The relative path and filename of the icon that appears to the left of the menu item. If you create any new icons, you should store them with the other icons, in the `Program Files\Honeywell\Experion PKS\Client\System\R`*xxx*`\Toolbar`, where *xxx* is the release number.

Finding the name of a Station command

The following example shows how to find the name of the command that is executed when a user chooses **Action > Request Report**.

To find the command’s name:

1 In Station choose **Station > Connection Properties**.

2 Click the **Toolbars** tab.

3 Click **Customize**.

4 Click the **Menu** tab. (If you were trying to find the name of a command associated with a toolbar icon or keyboard shortcut, you would click one of the other tabs.)

5 In **Current Layout** navigate to and select the “Request Report” entry.

The command’s name appears in **Command**.

Writing scripts for shortcut menus

Attention

Scripts for shortcut menus must be in JScript.

Types of script

There are two types of script that can be used with a menu item:

- onpage, which runs when the display is loaded
- onclick, which runs when a user clicks the menu item

Specialized properties and methods

The following properties and methods are specific to shortcut menus.

	Description
CloseShortCutMenu	This method closes the shortcut menu. It is used at the end of each script to close the menu.
customShortcutMenuSourceElement	This property represents the object right-clicked by the user.
InvokeShortcutMenu	This method invokes a shortcut menu. See “Invoking a shortcut menu from a display script” on page 135.
IsAccessPermitted	This method returns `True` if the user has access to the point linked to the object which was right-clicked. Otherwise, it returns `False`.
IsInAlarm	This method returns `True` if the point linked to the object which was right-clicked is currently in alarm and the alarm has not been acknowledged. Otherwise, it returns `False`.

Cancelling the event bubbling after a shortcut menu has been invoked

If you have invoked a shortcut menu with a script on the oncontextmenu event of a group, then you need to add the line:

```
window.event.cancelBubble=true
```

after calling InvokeShortcutMenu. Otherwise the default system shortcut menu will appear overriding the menu bought up via a script.

Configuring a s shortcut menu to call up an application

This example shows how to call up an application, such as the Alarm Configuration Manager.

```
<script type="onclick">
   var Parameter =
   window.external.parentWindow.document.activeElement;
   var App = "ACM.exe";
   var Parameters = App + " " + Parameter;
   window.external.application.Shell(Parameters, 1);
   window.external.CloseShortcutMenu();
   </script>
```

Referencing objects in the display

`window.external.parentWindow` is used to access the display and its objects.

This example shows how to shift an object, “rect001”, 10 pixels to the right.

```
<menuitem id="mnu_shift" acceleratorKey="">
  <script type="onclick">
    window.external.parentWindow.rect001.style.pixelLeft =
   window.external.parentWindow.rect001.style.pixelLeft + 10;
    return true
    window.external.CloseShortcutMenu();
  </script>
  <text>Shift rectangle</text>
  <image>.\Images\image1.bmp</image>
</menuitem>
```

Referencing objects and custom properties within a group or shape

Objects and custom properties within a group or shape are referenced in the normal manner.

This example shows how get the name of a file (stored in the custom property called “file1” on “shape001”), and then open the file in a new window.

```
<menuitem id="mnu_guide1" acceleratorKey="">
  <script type="onclick">
    var color =
   window.external.parentWindow.shape001.GetCustomProperty("text
   ", "color1");
window.external.parentWindow.alpha001.fillColor = color;
    window.external.CloseShortcutMenu();
  </script>
  <text>Operator Guide</text>
  <image>.\Images\image2.bmp</image>
</menuitem>
```

Referencing the Station Application object from the shortcut menu

`window.external.application` is used to access the Station Application object.

This example performs the same command as choosing **Control > Lower** from Station's menu. (If the object is not linked to a point, a message will appear in the Message Zone telling the user that a point is not selected.)

```
<menuitem id="mnu_grp" acceleratorKey="">
  <script type="onclick">
    window.external.application.InvokeMenu2('Lower');
    window.external.CloseShortcutMenu();
  </script>
  <text>Lower</text>
  <image>.\Toolbar\tb013.bmp</image>
</menuitem>
```

Disabling menu items based on security settings

You use an onpage script to conditionally enable or disable menu items.

You use the IsAccessPermitted method to enable/disable the menu item based on the security settings. (For a description of this and other specialized methods/properties, see "Specialized properties and methods" on page 130.)

This method is typically used in combination with the customShortcutMenuSourceElement property to determine whether the user has access to the object that was right-clicked.

This example shows a typical script.

```
<menuitem id="mnu_raise" acceleratorKey="">
  <script type="onclick">
    window.external.application.InvokeMenu2('Raise');
    window.external.CloseShortcutMenu();
  </script>
  <script type="onpage">
    if( !window.external.IsAccessPermitted(
   window.external.customShortcutMenuSourceElement )){
      this.disabled=true;
    }
    return true;
  </script>
  <text>Raise</text>
  <image>.\Toolbar\tb012.bmp</image>
</menuitem>
```

If the shortcut menu is not associated with the current object (that is, the object that has focus), you use `parentWindow.document.activeElement` to check whether the user has access to the object.

```
<menuitem id="mnu_raise" acceleratorKey="">
  <script type="onclick">
    window.external.application.InvokeMenu2('Raise');
    window.external.CloseShortcutMenu();
  </script>
  <script type="onpage">
    if( !window.external.IsAccessPermitted(
   window.external.parentWindow.document.activeElement )){
      this.disabled=true;
   }
     return true;
  </script>
  <text>Raise</text>
  <image>.\Toolbar\tb012.bmp</image>
</menuitem>
```

Disabling menu items based on other conditions

You can disable a menu item based on a range of conditions, other than the security settings.

This example disables the menu item if the current active object is “shape001” and the protocol is HTTP.

```
<menuitem id="mnu_raise" acceleratorKey="">
  <script type="onclick">
    window.external.application.InvokeMenu2('Raise');
    window.external.CloseShortcutMenu();
  </script>
  <script type="onpage">
    if( (window.external.parentWindow.document.activeElement.id
   == "shape001") &&
   (window.external.parentWindow.document.protocol == "http")){
      this.disabled=true;
    }
      return true;
  </script>
  <text>Raise</text>
  <image>.\Toolbar\tb012.bmp</image>
</menuitem>
```

Changing the size or style of a shortcut menu

When you create a shortcut menu, you may want to control the height and width to suit your content. To do this you use the top-level menu element in the shortcut menu.

This example shows how to specify the font and other characteristics of a shortcut menu.

```
<menu
  menuItemStyle="{font-family:sans-serif;font-size:9pt;padding-
   left:2;background-Color:menu;color:black; cursor:hand}"
  highlightItemStyle="{font-family:sans-serif;font-
   size:9pt;padding-left:2;background-
   Color:highlight;color:white; cursor:hand}"
  width="150"
  height="90">
```

Scripts not associated with a particular menu item

You can use a script not attached to a particular menu item to pass variables to particular menu items.

This example shows how to set the text of a menu item to the ID of the current object.

```
<?xml ... >
<menu ... >
  <script>
    var text=
   window.external.parentWindow.document.activeElement.id
  </script>
  <menuitem ....>
    <text>
      <![CDATA[.
        <script type="text/javascript" language="javascript">
          document.write(text);
        </script>
      .]]>
    </text>
  </menuitem>
  .
  .
</menu>
```

Invoking a shortcut menu from a display script

By default, a shortcut menu is invoked when the user right-clicks the associated object. However, scripts can also invoke a shortcut menu with the InvokeShortcutMenu method.

The following example invokes a shortcut menu, which appears at the current cursor location. (Note that the menu must be in one of Station's display paths in order to find it.)

```
Sub pushbutton001_onclick
  HDXPopupBehavior.InvokeShortcutMenu "myShortcutMenu.xml",
   window.event.x, window.event.y
End Sub
```

Optimizing display performance

5

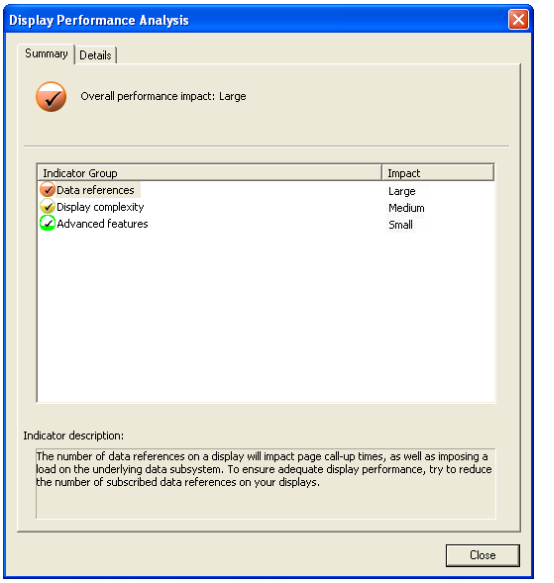
This chapter describes how to achieve the best performance for your displays.

Using the Display Performance Analysis tool

You can use the Display Performance Analysis tool to check a display's overall size and *performance indicators*, which include the number of objects and number of scripts. The summary page gives an indication of the overall size of the display based on Data References, Display Complexity and the use of Advanced Features.

For example, in the screenshot below, if the display developer wanted to improve the performance of their display, the Performance Display Analysis tool recommends minimizing the Data references first and then the Display complexity and finally, the Advanced features.

Figure 9 The performance summary for an example display



These categories represent a summary of the performance indicators listed on the Details page. If the display exceeds (or gets close to) the recommended limit for a particular indicator, you can read the associated notes that describe how to avoid exceeding it.

The recommended limits are not absolute, and are mainly provided as a reference to identify areas of potential improvement. Performance also depends on other factors such as the computer's capabilities (processor, speed and memory) and the number of applications it is running.

The analysis tool reports the number of build time data references in a display. At runtime, these data references map to a range of objects that have different performance characteristics. These include point references and database references.

The performance cost of a point reference varies depending on the type of reference and where the data is being retrieved from. Point references include:

- Controller data

  Values must be retrieved from the controllers connected to the server or Console Station. The performance cost of these references depends largely on the characteristics of the underlying controller.

- Cache data

  Some point values, such as name, are permanently cached, other values are temporarily in cache because of previous client requests. The cost of referencing a cached value is quite low. Duplicate references are also filtered out to minimize the load on the controller.

- Indirect data

  Properties such as alarm state and quality are derived from other information being maintained by the server. These references compete for resources on the server and can cause more loading than a normal database reference.

- DSA data

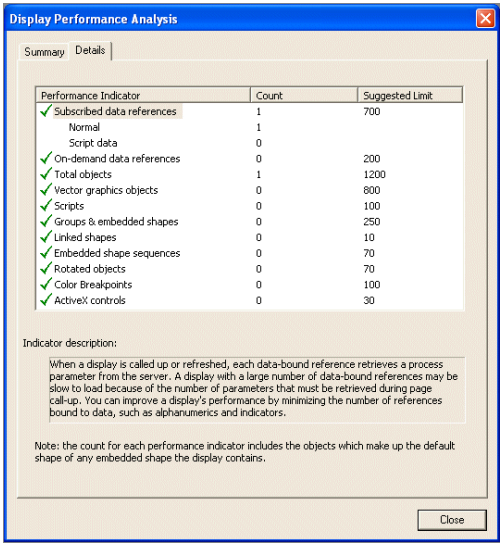
  Referencing point data from another Experion server will slightly impact performance compared to referencing the same data locally.

Database references cause the least load on the system as the database holds some information about the system, for example, trend configuration data.

HMIWeb Display Builder does not differentiate between these data reference types but you should be aware of the varying impacts and use your knowledge of the system to consider the server loading caused by the different types of references on your displays.

Improving a display's performance is an iterative task that involves repeatedly using the tool to identify each performance indicator that adversely affects performance, and then making the appropriate changes to improve that indicator.

Figure 10 The performance indicators for an example display



Notes

- The counts do not include items within linked shapes (but they do include items within embedded shapes). This means that if you embed a previously linked shape, most counts will increase and some counts may become excessive even though the display's performance will be improved.

To analyze a display's performance:

1. If several displays are open, select the one you want to analyze.
2. Choose **Tools** > **Display Performance Analysis**.

   The Display Performance Analysis tool shows a summary of the display.
3. Click the **Details** tab to view the performance indicator results.
4. If there is a problem with any indicator, select it to see the associated notes.

Display-creation guidelines

This section includes guidelines to help you achieve the best performance for your displays.

Minimizing the object count

The processing required by Station to call up a display depends primarily on the object count, almost regardless of the complexity of individual objects.

You can use the following techniques to reduce the object count, without affecting the display's appearance:

- Use the **Combine** tool ( ) to convert static objects, such as lines and rectangles, into a single bitmap. See "Combining objects to improve display performance" on page 83.
- Use the **Polyline** tool to draw a single, complex line, rather than using the **Line** tool to draw a series of straight lines.
- Avoid excessive use of the **Group** tool (for example, nesting groups with groups) because each time you use the tool, you create an additional group object.
- Do not use invisible alphanumerics to enable scripts to access point parameter values. Instead, add the point parameters to the **Script Data** tab of a suitable object. To learn how to write scripts using this technique, see "Reading and writing to point parameters" on page 304.
- Use a third-party package to create separate graphics for complex static components, such as backgrounds, and then inserting them into the display. Such graphics are treated as single objects, regardless of their internal complexity. Suitable graphic formats are:
  - WMF (Windows MetaFile) and EMF (Enhanced MetaFile). These are vector formats that result in small files that scale well.
  - PNG. This is best for bitmap graphics with transparent backgrounds.
  - JPEG. This is best for photographic-type images.

Note that the above techniques are only necessary if the display is particularly complex, or if it contains numerous shapes.

If you are preparing a shape library for general usage, you should pay particular attention to the above techniques, so that the performance of displays that make extensive use of your shapes is not compromised.

Minimizing the number of active objects

You can improve a display's performance by minimizing the number of *active* objects, such as alphanumerics and indicators or animations.

This also has usability advantages because, from a user's point of view, simple and uncluttered displays are generally easier to use.

When a display is called up or refreshed, each active object retrieves multiple process parameters from the server—even a simple object such as an

alphanumeric, requires about five parameters, such as Value, Alarm State and Scan Status. Consequently, a display with a large number of active objects may have performance problems because of the number of parameters that must be retrieved.

When using script data, if the point-related values are not required immediately after a display is called up, you can improve the display's callup performance by setting the DataChanged property of those points to `On Demand`. You would then use the RequestDemandData method to obtain values when required. See "Properties and methods of points on the Script Data tab" on page 306.

Using shapes

You should use the shapes supplied with HMIWeb Display Builder wherever possible because they have been optimized for display performance. See “Shape library” on page 41.

Apart from improving display performance, the supplied shapes make it easier to create visually consistent displays.

Embedding shapes

Embed shapes where possible by selecting the **Contents of shape file** and clearing the **Link to shape file** option when you insert a shape. See “Inserting a shape sequence” on page 96 and “Inserting a dynamic shape” on page 97.

If a display already contains linked shapes, you can embed them by selecting them and then choosing **Tools > Embed Shapes**.

Avoid reloading shapes

Avoid reloading shape objects, particularly those which contain data-bound elements. The use of the following methods and properties cause a shape to be reloaded, and should therefore be minimized:

- Shape object SetCustomProperty() method
- Shape object ReloadContent() method
- Shape object ID property

If you want to set several custom properties, you can improve the performance by first setting the shape’s autoReloadContent property to False, then setting the custom properties, and finally using the ReloadContent method. For example:

```
Sub pushbutton001_onclick
  ' Prevent the shape being updated as each custom property is
   changed
  shape001.autoReloadContent = false
  ' Set the custom properties
  shape001.SetCustomProperty "Point", "Point1", "POINTANA1"
  shape001.SetCustomProperty "Parameter", "Param1", "PV"
  shape001.SetCustomProperty "Point", "Point2", "POINTANA2"
  shape001.SetCustomProperty "Parameter", "Param2", "PV"
  ' Apply the changes to the shape
  shape001.ReloadContent
End Sub
```

Creating your own shapes

If you need to create your own shapes, pay particular attention to the object count and scripts, so that the performance of a display that makes extensive use of your shapes is not compromised.

Avoid nesting shapes

Avoid inserting a shape within another shape.

Minimizing the use of custom properties

Minimize the use of custom properties within shape sequences.

Minimizing the use of scripts

Minimize the use of scripts within shape sequences. If you include scripts, pay particular attention to the guidelines for scripts.

Minimizing the number of lines in a script

The performance of a single-line script (including comments) is much better than an equivalent multi-line script. In VBScript, you use a colon (:) to separate multiple statements on a single line, for example:

```
Statement 1: Statement 2: Statement 3: 'Comment
```

In the case of shapes, you should also organize the scripts so that each event handler refers to one or more functions in the general section, either within the shape or the display in which it is inserted. For example:

```
Call Function A: Call Function B
```

For more details see the section “Using general section functions” on page 296.

Minimizing the use of behaviors

Deselect any object behaviors that are not required—some behaviors, such as color breakpoints, involve additional processing even if they are not used.

Using scripts

Minimizing the number of scripts

The performance of a single, long script is better than several short scripts. For example, try writing a single, long script on one object event rather than several short scripts on several object events.

Instead of writing a script to create a custom behavior for an object, consider using a standard behavior such as acronyms, shape sequences, level fills or color breakpoints.

Referencing objects

You should avoid using the document.all and Page.all collections to reference objects as they slow down the display. Instead, it is recommended that you use the document.getElementByID() method or reference the objects directly by name in the script. If you need to reference an object within a group, using the **all** or **children** collections available on the group object is the recommended approach.

Errors in scripts

VBScript allows you to use on the “on error resume next” statement to hide any errors from the users. However, any errors that are triggered within scripts have a significant performance cost on the display. You should correct the script errors rather than hiding them using the “on error resume next” statement.

On demand script data

Scripting is an essential part of any HMIWeb display and using scripts correctly can optimize the performance of a display.

On demand scripts can be used for events that are only needed be run when demanded, such as the display of tooltips, rather than run every time the display is opened.

For example, the following script can be placed on a shape to invoke a tooltip displaying the value of a sinewave.

```
Sub shape003_ondatachange

if (shape003.DataChanged("On Demand")) then
   shape003.title = "Value: " &
   shape003.DataValue("sinewave.pv")
end if

End Sub
```

The following script is added to the onmouseover event of the shape:

```
Sub shape003_onmouseover
shape003.RequestDemandData
End Sub
```

The on demand scripts are well suited for information that the operator does not always have to see, such as alarm counts or units in large displays. The on demand scripts save CPU resources and improves the performance of a display.

Using InvokeTask and RequestServerLRN functions in scripts

Display scripting functions such as InvokeTask and RequestServerLRN can invoke task requests on the server or console Station. These are costly functions that slow down the display. These functions should be avoided during display call-up.

Using breakpoints

Using breakpoints gives you more flexible control of the display element behavior without having to resort to display script code, which results in improved performance.

Miscellaneous guidelines

Minimizing the use of ActiveX controls

ActiveX controls take slightly longer to render than native HMIWeb objects.

You should therefore avoid using large numbers of ActiveX controls.

Sizing displays to fit Station's setup

Display performance is best if the display is rendered in Station at 100% magnification, without scrollbars.

If you have a standard Station setup—Station is maximized and the monitor's resolution is set to `1024x768` pixels—use HMIWeb Display Builder's default display size (`1020x589` pixels).

However, if Station monitors use a different resolution, you should consider creating displays optimized for that resolution. As a starting point for determining the optimum display size, create a display that is slightly smaller than the monitor's resolution.

Configuring display properties

If you find that you are experiencing a high CPU usage with your displays, you should ensure that the Display Properties Control Panel on the node are configured to:

- Medium (16 bit) color
- Maximum screen resolution of 1280x1024.

Station-related guidelines

If your displays include linked shapes, you can improve Station's performance by clearing the **Search subdirectories for shapes** check box on the **Connection Properties** dialog box (**Displays** tab), and then explicitly defining each subdirectory that contains displays.

This is especially important if you store displays in a centralized location on the network.

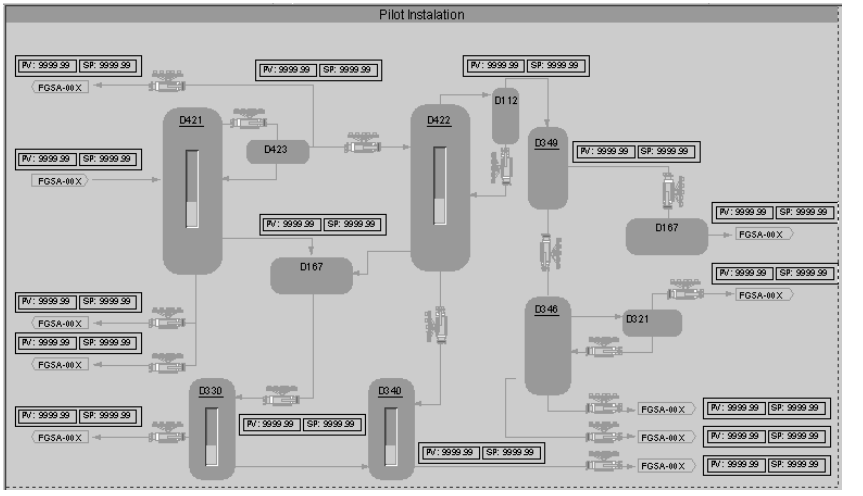
For details, see the *Configuration Guide*.

Improving the performance of an existing display (an example)

This example describes how to dramatically improve the performance of an existing display.

The following display includes numerous static objects (lines, tanks, and so on), as well as numerous links to a dynamic shape (the readouts) and a static shape (the valves).

Figure 11 The display

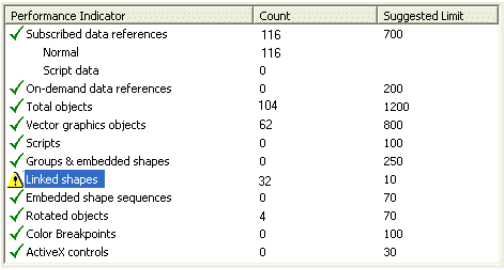


As shown in the following procedure, improving the display's performance is an iterative task that involves repeatedly using the Display Performance Analysis tool to identify a performance indicator that adversely affects performance, and then making the appropriate changes to improve that indicator.

To improve the display's performance:

1 Use the Display Performance Analysis tool to check the display's performance indicators. See "Using the Display Performance Analysis tool" on page 138.

Figure 12 The performance indicators for the original display



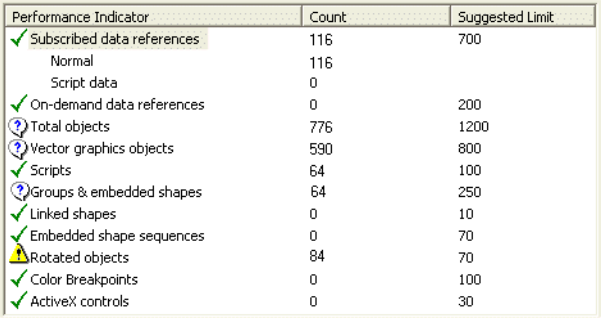
Performance Indicator	Count	Suggested Limit
Subscribed data references	116	700
Normal	116	
Script data	0	
On-demand data references	0	200
Total objects	104	1200
Vector graphics objects	62	800
Scripts	0	100
Groups & embedded shapes	0	250
Linked shapes	32	10
Embedded shape sequences	0	70
Rotated objects	4	70
Color Breakpoints	0	100
ActiveX controls	0	30

2 Because the number of linked shapes exceeds the recommended limit, you embed them. See “Embedding a linked shape” on page 98.

3 Use the Display Performance Analysis tool again.

This time note how the counts for some indicators, such as vector graphic objects, have increased. (The tool does not count objects within linked shapes—so embedding a shape will increase some counts.)

Figure 13 The performance indicators after embedding the shapes



Performance Indicator	Count	Suggested Limit
Subscribed data references	116	700
Normal	116	
Script data	0	
On-demand data references	0	200
Total objects	776	1200
Vector graphics objects	590	800
Scripts	64	100
Groups & embedded shapes	64	250
Linked shapes	0	10
Embedded shape sequences	0	70
Rotated objects	84	70
Color Breakpoints	0	100
ActiveX controls	0	30

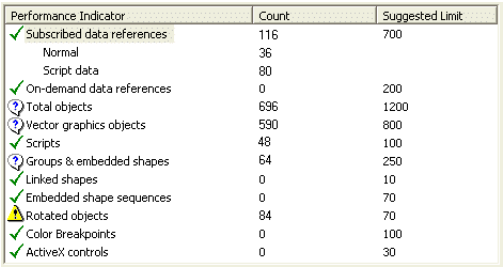
4 Because there are a large number of scripts and data-bound object (dynamic objects), which you know belong to a particular dynamic shape, you decide to improve the shape’s design. See “Improving the design of the dynamic shape” on page 156.

5 Upgrade the embedded shapes that are based on this shape. See “Upgrading an embedded shape” on page 99.

6 Use the Display Performance Analysis tool again.

This time note how the count for scripts has been significantly reduced. In addition, the data bound object has been redistributed so that most of the data is being received by way of script data. Script data is the most efficient way to use process data in your displays, so this redistribution will improve the display load time.

Figure 14 The performance indicators after redesigning the dynamic shape



Performance Indicator	Count	Suggested Limit
Subscribed data references	116	700
Normal	36	
Script data	80	
On-demand data references	0	200
Total objects	696	1200
Vector graphics objects	590	800
Scripts	48	100
Groups & embedded shapes	64	250
Linked shapes	0	10
Embedded shape sequences	0	70
Rotated objects	84	70
Color Breakpoints	0	100
ActiveX controls	0	30

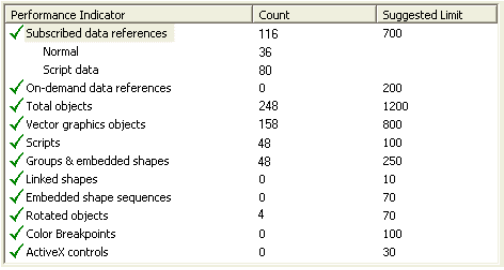
7 Because you know that the static shape (the valve) includes many objects, you decide to combine its objects to create a single bitmap. See “Combining objects to improve display performance” on page 83.

8 Upgrade the embedded shapes that are based on this shape. See “Upgrading an embedded shape” on page 99.

9 Use the Display Performance Analysis tool again.

This time note how the count for Total objects has is less than half what it was.

Figure 15 The performance indicators after redesigning the static shape



Performance Indicator	Count	Suggested Limit
Subscribed data references	116	700
Normal	36	
Script data	80	
On-demand data references	0	200
Total objects	248	1200
Vector graphics objects	158	800
Scripts	48	100
Groups & embedded shapes	48	250
Linked shapes	0	10
Embedded shape sequences	0	70
Rotated objects	4	70
Color Breakpoints	0	100
ActiveX controls	0	30

10 Because the display’s background includes many vector graphic objects, you decide to combine them to create a single bitmap. See “Combining objects to improve display performance” on page 83.

11 Use the Display Performance Analysis tool again.

This time note how the count for Total objects has been further reduced.

Figure 16 The performance indicators after redesigning the static shape

Performance Indicator	Count	Suggested Limit
✓ Subscribed data references	116	700
Normal	36	
Script data	80	
✓ On-demand data references	0	200
✓ Total objects	203	1200
✓ Vector graphics objects	113	800
✓ Scripts	48	100
✓ Groups & embedded shapes	48	250
✓ Linked shapes	0	10
✓ Embedded shape sequences	0	70
✓ Rotated objects	4	70
✓ Color Breakpoints	0	100
✓ ActiveX controls	0	30

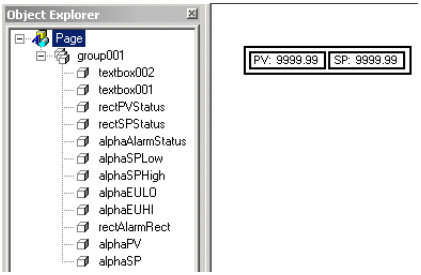
Improving the design of the dynamic shape

The dynamic shape contains 13 objects, including five invisible alphanumerics which provide access to point parameter values.

The recommended way of accessing parameter values is to add them to the **Script Data** tab of a suitable object. (This means that you no longer need the five invisible alphanumerics.)

In this example, you add them to the group object. (Note that you have to modify the shape's scripts so that they use the new syntax to access the parameters.)

Figure 17 The original shape, with its objects shown in the object explorer



To improve the shape's performance:

1. Add the required point parameters to the Script Data tab of the group object. See "Script Data tab" on page 201.
2. Modify the scripts so that they use the new syntax access the parameter. The following figure shows the original and updated versions of a typical script.

Figure 18 A typical script (original above, updated below)

```
Sub alphaSP_onupdate

    If (alphaSP.value >= alphaSPHigh.value) Then
        rectSPStatus.lineColor = window.external.MakeColor(255,0,0)
    ElseIf (alphaSP.value <= alphaSPLow.value) Then
        rectSPStatus.lineColor = window.external.MakeColor(0,0,255)
    Else
        rectSPStatus.lineColor = window.external.MakeColor(0,255,0)
    End if

End Sub
```

```
Sub alphaSP_onupdate

    If (alphaSP.value >= group001.DataValue("sinewave.SetPointHighLimit")) Then
        rectSPStatus.lineColor = window.external.MakeColor(255,0,0)
    ElseIf (alphaSP.value <= group001.DataValue("sinewave.SetPointLowLimit")) Then
        rectSPStatus.lineColor = window.external.MakeColor(0,0,255)
    Else
        rectSPStatus.lineColor = window.external.MakeColor(0,255,0)
    End if

End Sub
```

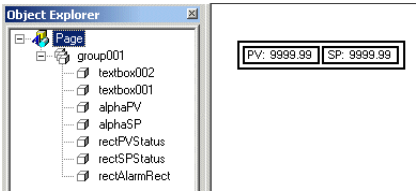
The following table shows the differences between the original and updated syntax used to access the same parameter. (In the original code, the alphanumeric would have been linked to the SetPointHighLimit parameter of point called “sinewave”.)

Original code	Updated code
`alphaSPHigh.value`	`group001.DataValue("sinewave.SetPointHighLimit")`

3 Delete the invisible alphanumerics.

Note how the shape now only contains eight objects.

Figure 19 The updated shape, with its objects shown in the Object Explorer



Optimizing the performance of shapes containing scripts

If you have shapes that contain scripts repeated on a display, you should ensure the scripts are a single line to optimize the performance of the display.

One method you can use to create one line scripts is to create a special shape called a “GeneralScriptHolder”. This shape is then made invisible so that it is not displayed in Station. Each shape then calls methods stored in the GeneralScriptHolder.

Below is an example of a script that can be added into the General section of the script editor for the GeneralScriptHolder shape:

```
‘-----------------------------
CDA Alarm enable state
‘-----------------------------
sub CDA_Alarm_AlmEnb_OnUpdate(oSource)
dim oShape
On error resume next
set oShape=oSource.parentelement.parentelement
if oSource.Datavalue(“tagname.cp_almenbstate”) = 0 then
oShape.objects(“TxtAlmEnbState”).style.visibility = “visible”
oShape.objects(“ScConAlarmState”).style.visibility = “hidden”
else
oShape.objects(“TxtAlmEnbState”).style.visiblility = “hidden”
oShape.objects(“ScConAlarmState”).style.visibility = “visible”
end if
end sub
```

In each of the repeated shapes on the display enter a script such as:

```
SubScConAlarmState_ondatachange
call HPM_Alarm_AlmEnb_OnUpdate(me)
End Sub
```

You will notice that the object passes a reference to itself to the procedure so that the properties can be changed. This means that each shape doesn’t need to have a copy of the block of script. Repeating the script block in a number of shapes significantly slows the loading of the page.

The aim is always to have a single line of script in each handler. Having only one line allows the storage of the scripts to be optimized even more. A small number of scripts can be placed on the same line separated by colons if needed. An added benefit of this approach is that a library of scripts are built up in one location rather than spread throughout several different shape files. This encourages the re-use of existing routines and also simplifies maintenance.

When using scripts, it is always best to try and do the least amount of work as possible to achieve the result you want. For example, if you had 60 shapes on a page with the onupdate handler:

```
IK_num=me.parentElement.parentElement.getcustomproperty("VLAUE",
   "IK_Number")

dim Ret, Rev, dis, tmfwd, tmrev, hiamps

Ret = link.Datavalue ("Retraced_tag.pvfl")
Rev = link.DataValue ("Reversed_tag.pvfl")
dis = link.Data.value ("Disabled_tag.op")
tmfwd = link.Datavalue ("Elapsed_fwd_tag.pvfl")
tmrev = link.Datavalue ("Elapsed_fwd_tag.pvfl")
hiamps = link.Datavalue (Amps_high_tag.pvfl")

dim ovrld

ovrld = link.Datavalue ("OverLoad_tag.pvfl")

'magenta no blink
If dis = 1 then
IK_num.textColor = window.external.MakeColor(255,0,255)
IK_num.textColorBlink = false

'magenta blink
elseif dis = 1 and ret = 0 then
IK_num.textColor = window.external.MakeColor(255,0,255)
IK_num.textColorBlink = True

...
```

Every time the display updates all the shapes us select the six data values and then potentially use only one or two of them. It would be better to only ask for the data values as it becomes apparent that they are actually needed. For example, if dis = 1, then five values are requested but never used. Using the principle of only asking for the values that you need to minimize the work done by your scripts.

Shapes sequences are quite slow, so if you are going to use a shape sequence many times on a display, then it would be more efficient to change the appearance of shapes using scripts, for example showing and hiding rectangles and circles. This will ensure that the performance of the shapes is optimized, as long as you follow the recommendation to ensure that the shapes have only one line of script per handler.

You should regularly test the callup and steady state performance of displays as they are built to get an awareness of how different objects affect the performance of a display.

Object properties

6

This chapter describes the object properties.

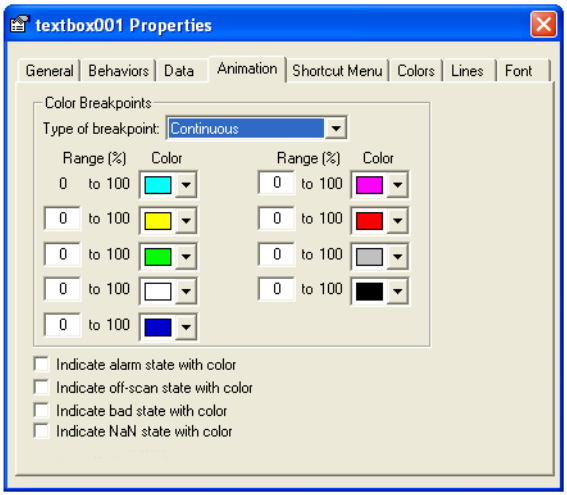
You access the properties of the selected object(s) through the Properties Window. (If the Window is not visible, click on the View toolbar.)

For details about this tab:	Go to:
Animation	page 163
Behaviors	page 165
Colors	page 167
Columns (table)	page 169
Custom Properties	page 168
Data	page 170
Data (alarm or event table)	page 175
Details	page 176
Filters	page 193
Font	page 194
General	page 195
Layout (table)	page 197
Lines	page 197
Plots	page 198
Popup	page 199
Repeat (table)	page 200
Script Data	page 201
Scrollbar	page 204
Shortcut Menu	page 207
Sorting	page 208
Period	page 209

For details about this tab:	Go to:
View	page 211

Animation tab

The Animation tab specifies the correspondence between the object's color and the value or status of the associated point parameter/database file.



Property	Description
Color Breakpoints	The type of animation used for the object: • `None`. No animation. • `Continuous`. The color changes as the value changes. You can specify up to nine colors and the percentage range for which each color applies. (For example, you could make an object turn red when the value is between 90 and 100 percent of its range.) • `Discrete`. Each color represents a particular discrete value. Select **Inverse** if you want to use the object's fill color—this produces a "reverse video" effect, which makes text stand out, providing you have selected an appropriate color scheme.
Indicate alarm state with color	Station overrides the current color with the appropriate system-defined color when the associated point is in alarm. See "System-defined colors" on page 164.
Indicate off-scan state with color	Station overrides the current color with the system-defined color for "off-scan". (See "System-defined colors" on page 164.)

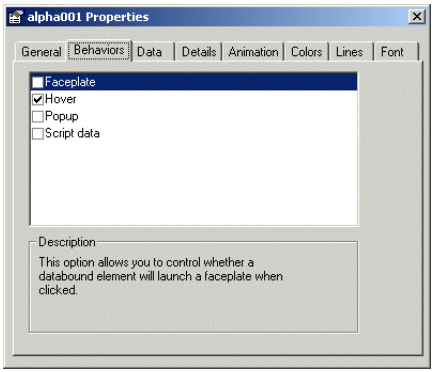
Property	Description
Indicate bad value with color	Station overrides the current color to indicate a bad value, such as a bad quality, OOS, or a lost value. The bad state color is defined in the station.ini file.
Indicate NaN with color	Station overrides the current color with the system-defined colors to indicate the bound data's NaN quality state. The NaN state color is defined in the station.ini file.
Level fill animation	Only applicable to vector objects, such as ovals. The "level" of the object's fill color varies in accordance with a value in the database. Specify the database details on the Data tab.

System-defined colors

Point state	System-defined color
In alarm and unacknowledged	Red and blinking
In alarm and acknowledged	Red
Returned to normal but alarm not acknowledged	Normal color, but blinking
Point off-scan (scanning disabled)	White

Behaviors tab

The Behaviors tab specifies which behaviors are attached to the object.



Property	Description
List	Shows the behaviors that are available for the object. For a description of behaviors supplied with HMIWeb Display Builder, see “Standard behaviors” on page 165.
Description	Provides a brief description of the highlighted behavior.

Standard behaviors

The following behaviors are supplied with HMIWeb Display Builder. (Note that some behaviors are not applicable to all objects.)

Behavior	Description
Faceplate	A *faceplate* appears when a user clicks the object. A faceplate is a specialized type of popup that shows critical information about the point. (In most cases, a faceplate looks like the left-hand portion of the associated point detail display.) To learn about popups, see “About popups and faceplates” on page 59.

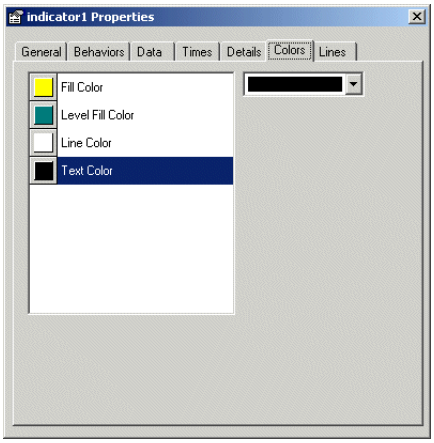
Behavior	Description
Hover	Only applicable to alphanumerics and comboboxes. If the object is too narrow to show its contents, the contents appears in a ToolTip (a small window) when a user moves the mouse pointer over the object. **Note:** If you select this behavior, do not use the object's **ToolTip** property (General tab).
Popup	A popup appears when a user clicks the object. Adds the Popup tab, in which you specify the popup's properties. To learn about popups, see "About popups and faceplates" on page 59.
Script Data	Enables scripts to access point parameter values. Adds the Script Data tab, in which you specify the point parameters that can be accessed by scripts. See "Reading and writing to point parameters" on page 304.
Shape Sequence Animation	Only applicable to shape sequences. See "Inserting a shape sequence" on page 96.

Colors tab

The Colors tab controls the object's colors.

Notes

- The number of color properties depends on the object's type. For example, a rectangle has line and fill colors, whereas an alphanumeric also has a text color. (The following table only describes common elements.)



Property	Description
Fill Color	The object's fill color.
Line Color	The object's line color.
Text Color	The text color. Only applicable to objects with text, such as alphanumerics and pushbuttons.
Level Fill Color	The color of the bar or pointer used to indicate the value. Only applicable to indicator objects.
Trend-specific properties	
axisBackColor	The background color of the axes.
AxisLabelColor	The color of the axis labels.
AxisMarkerLabelColor	The color of the axis markers.
plotBackColor	The background color of the plot.
referenceLineColor	The color of the reference line.

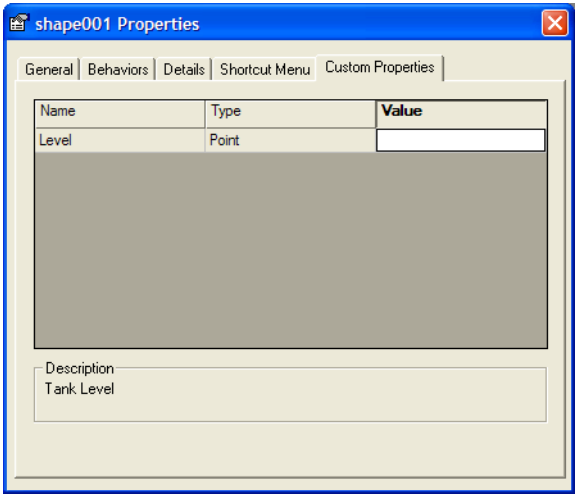
Custom Properties tab

The Custom Properties tab is available when you insert a shape into a display. You can set the value for the custom property on this tab. (Custom properties are added in the shape file.)

Notes

- There are two types of custom properties, shape custom properties and system custom properties. This topic refers to shape custom properties. For information about system custom properties, see “System Custom Properties tab” on page 247.

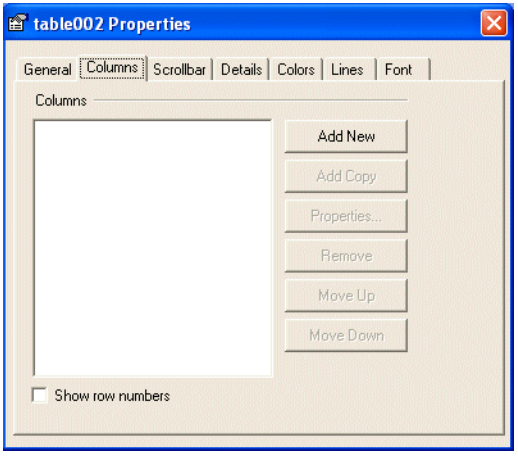
To learn:	Go to:
About custom properties	page 57
How to use this tab when inserting a shape sequence	page 96
How to use this tab when inserting a dynamic shape	page 97



Columns tab (table)

The Columns tab is available when you insert a table into your display. It is used to add, delete, and modify columns in the table.

To learn:	Go to:
About tables	page 57
Creating a table	page 241
Modifying a table	page 242



Data tab

The Data tab specifies the object's link to the database. (Such objects are called *dynamic objects*.)

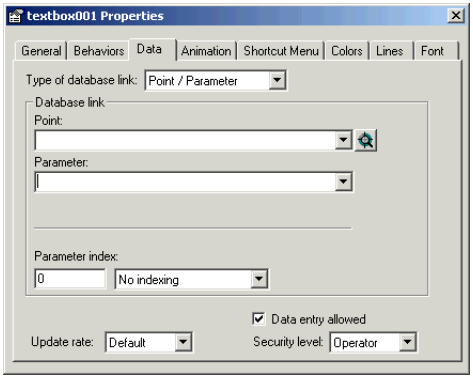
For some types of objects, such as rectangles and circles, this tab only appears if you select **Color Animation** on the Behaviors tab.

The **Type of database link** property specifies the type of link:

- `None`. The object is not linked to anything. (In the case of a combobox, for example, you would select this if you wanted to populate the combobox using a script.)
- `Point/Parameter`. The object is linked to a point parameter. See "Defining a link to a point" on page 171.
- `Database file`. The object is linked to a file in the server database, whose contents are controlled by another application. See "Defining a link to a database file" on page 173.

Defining a link to a point

When defining a link to a point, you specify the point and the appropriate parameter. (Each point has a number of parameters associated with it, such as field values and configuration details. For details about points and their parameters, see the *Configuration Guide*.)

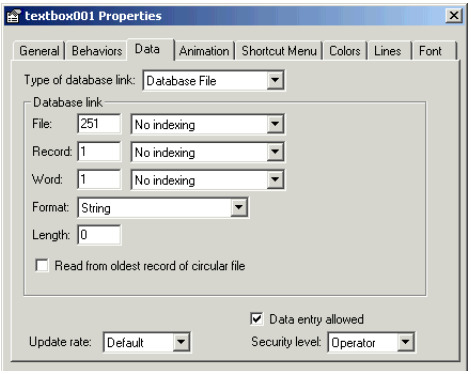


Property	Description
Point	The ID of the point or the full item name to which the object is linked. If you are creating the display on a server, you can select the tag name or full item name from the list. (However, in a DSA system, the server you are working on may not know of the existence of every point in the system.) You can click the Point Browser button to browse points on this server or another server in the system. See "Point Browser" on page 36.
Parameter	The point parameter to which the object is linked. If you select a history parameter, such as H24H, **offset** controls which history value is used. (`0`, the default, represents the latest history value, `1` represents the second-to-last value, and so on.) By default, history parameters refer to the history for the PV point parameter. To show the history for a point parameter other than PV, append a space character and the name of the required parameter to the history parameter. For example, to show the 1 minute history of PIDA.PV, type **`H1M PIDA.PV`**.

Property	Description
Parameter index	Select the parameter index: • **No indexing**. The parameter has no index. • **Indexed by field offset**. If the parameter has an index, select this value and type the index value in the text box.
Update rate	The frequency that the parameter value is updated: • `Default`. The value is updated at the Station update rate. • `Once-Off`. The value is only updated when the display is initially called up or refreshed. • `1 second`, `2 second`, `4 second`, `5 second`, `10 second`, `30 second`, and `1 minute`. **Note:** The actual update rate is also determined by the Station and display update rates. See "Actual update rate" on page 202.
Data entry allowed Security level	If you select **Data entry allowed**, users with a security level equal to or higher than the specified **Security Level** can edit the object's value. The security levels are: `Operator`, `Supervisor`, `Engineer` and `Manager`.

Defining a link to a database file

For details about database files, see the *Application Development Guide*.



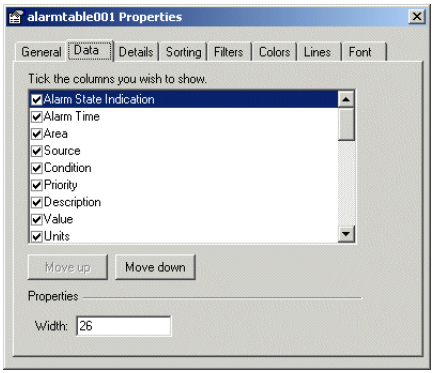
Property	Description
File	The number of the file to which the object is linked. Select the indexing option from the list opposite: • `No indexing`. • `Indexed by file offset`. Adds Station's current file offset to the file number. The offset can be set by scrollbars, by the host and by the PAGE UP and PAGE DOWN keys.
Record	The number of the record within the file. Select the indexing option from the list opposite: • `No indexing`. • `Indexed by record offset`. Adds Station's current record offset to the record number. The offset can be set by scrollbars, by the server and by the PAGE UP and PAGE DOWN keys. • `Indexed by field offset`. Adds Station's current field offset to the word number. The offset can be set by scrollbars, by the server and by the PAGE UP and PAGE DOWN keys. • `Indexed by Station no`. Each Station uses separate records in the user file. The effect is to view the user file as a series of logical files, one for each Station. Set the record and word numbers as normal; for example, if you want the link to refer to the second record for each Station, set **Record** to `2`.

Property	Description
Word	The word number within the file. Select the indexing option from the list opposite: • `No indexing`. • `Indexed by field offset`. Adds Station's current field offset to the word number. The offset can be set by scrollbars, by the server and by the PAGE UP and PAGE DOWN keys.
Format	The data format. If you select: • `String`, you must specify the length of the string. • `Bits`, you must specify the starting bit and the number of bits within the word.
Read from oldest record of circular file	Applicable if it is a circular file and you want to read the oldest record first.
Update rate	The frequency that the parameter value is updated: • `Default`. The value is updated at the Station update rate. • `Once-Off`. The value is only updated when the display is initially called up or refreshed. • `1 second`, `2 second`, `4 second`, `5 second`, `10 second`, `30 second`, and `1 minute`. **Note:** The actual update rate is also determined by the Station and display update rates. See "Actual update rate" on page 202.
Data entry allowed Security level	If you select **Data entry allowed**, users with a security level equal to or higher than the specified **Security Level** can edit the object's value. The security levels are: `Operator`, `Supervisor`, `Engineer` and `Manager`.

Data tab (alarm or event table)

The Data tab controls what columns (alarm/event details) appear in the table.

To learn about alarm/event tables, see “About alarm and event tables” on page 237.



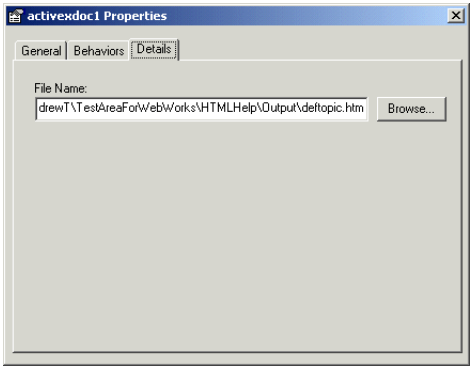
Property	Description
Tick the columns you wish to show	Lists the available columns that can appear in the table. Select the columns you want to appear in the table. Use the **Move up** and **Move down** buttons to specify the order in which columns appear in the table. (The first column in the list appears on the left of the table.)
Width	The width of the selected column. The way in which the width is interpreted depends on the setting for **Column resizing** (on the Details tab). If you set **Column resizing** to `None` or `Standard`, the width represents pixels. Note that if the sum of the column widths is greater than the width of the table (**Width** on the General tab), you should enable horizontal scrollbars (set **Horizontal scrollbar** on the Details tab to `visible`). If you set **Column resizing** to `Compact`, the width is treated as proportional value, and the column widths are automatically adjusted to fit the table. For example, if a table is 450 pixels wide and you set the width of the first column to 100 and the width of the four other columns to 200, the width of the first column will be 50 pixels (one ninth of 450) and the width of each other column will be 100 pixels (2/9 of 450).

Details tab

The Details tab contains properties that are specific to each type of object.

For details about this type of object:	Go to:
ActiveX control	page 177
Alarmstate	page 178
Alarm table	page 179
Alphanumeric	page 181
Checkbox	page 183
Combobox	page 184
Event table	page 179
Hyperlink	page 185
Indicator	page 186
Picture	page 187
Pushbutton	page 188
Shape	page 190
Table	page 191

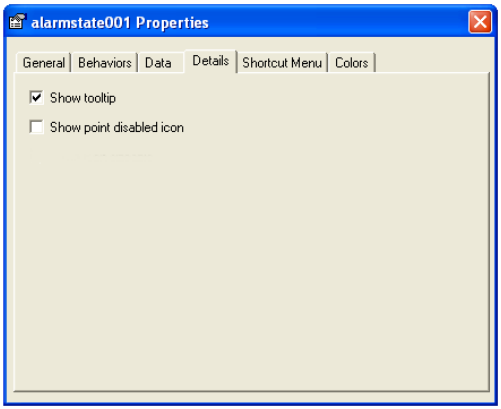
ActiveX document



Property	Description
File Name	The URL (Uniform Resource Locator) of the document that you want to display. If necessary, find the document by clicking the **Browse** button.

Alarmstate

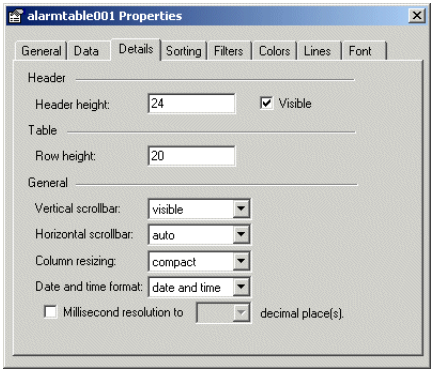
To learn how to set alarm states, see “Setting alarm states” on page 239.



Property	Description
Show tooltip	If this option is selected a tooltip will be displayed when a user hovers the mouse over the alarm state object on a display.
Show point disabled icon	Displays the point disabled icon D when a point has been disabled.

Alarm or event table

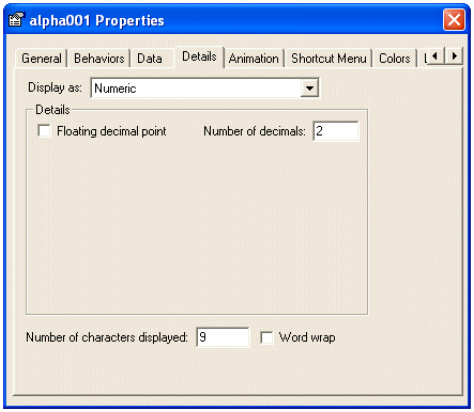
To learn about alarm and event tables, see “About alarm and event tables” on page 237.



Property	Description
Header	
Header height Visible	If **Visible** is selected, the table includes column headings. **Header height** specifies the height (in pixels) of the column headings.
Table	
Row height	The height (in pixels) of the rows.
General	
Vertical scrollbar	The vertical scrollbar’s visibility setting. The values are: • `Hidden` = the scrollbar is hidden • `Visible` = the scrollbar is visible • `Auto` = the scrollbar only appears if the table cannot simultaneously show all the alarms.
Horizontal scrollbar	The horizontal scrollbar’s visibility setting. The values are: • `Hidden` = the scrollbar is hidden • `Visible` = the scrollbar is visible • `Auto` = the scrollbar only appears if the table is narrower than total width of the columns.

Property	Description
Column resizing	Specifies whether a user can resize columns, and the way in which adjacent columns are resized: `None` = The columns cannot be resized `Standard` = Resizing a column makes the column to the right wider/narrower `Compact` = Resizing a column makes all columns to the right proportionally wider/narrower.
Date and time format	Specifies the date and time format. The formats are: • `Date and time` = shows the date, and time to seconds • `Time` = shows the time to seconds If you want greater than 1-second resolution, select the **Millisecond resolution to** option and specify the number of decimal places.

Alphanumeric



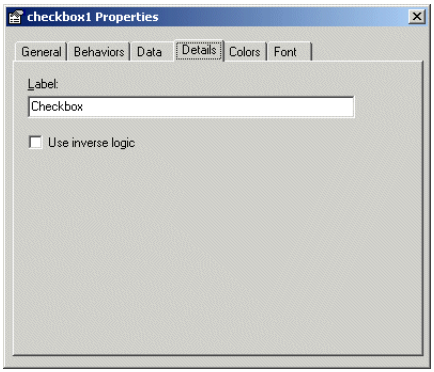
Property	Description
Display as Details	These properties specify how the value is displayed. See “Alphanumeric display options” on page 181.
Word wrap	If selected, the value (normally text) is displayed over several lines if it is longer than the alphanumeric’s width.
Number of characters displayed	The maximum number of characters that can be displayed.

Alphanumeric display options

Display As option	Description
Numeric	Displays the value as a number. If you clear the **Floating decimal point** check box, you must type the number of decimals you want to display in **Number of decimals**.
State Descriptor	Applicable to a status point. Displays the state descriptor associated with the current value.
Text	Applicable if the value is a text string. (255 characters maximum.)
Acronym	Displays the acronym associated with the current value (an integer) of the point parameter. See “Selecting acronyms” on page 219.

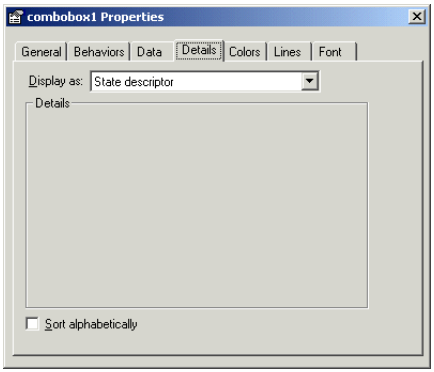
Display As option	Description
Time	Applicable if the value represents time. If the value is an integer, then it is assumed to represent minutes after midnight. If it is a real number, it is assumed to represent seconds after midnight. Specify the number of characters you want to display in **Decimals**.
Date	Applicable if the value is a date in Julian format.
Date/Time	Applicable if the value date and time.
Hexadecimal	Applicable if you want to display the value in hexadecimal format.
Source Address	Applicable if the value is a source address for a point. This option is generally only used by Honeywell personnel.
Destination Address	Applicable if the value is a destination address for a point. This option is generally only used by Honeywell personnel.
Parameter of given point	Applicable if the value is a point number in a user file. Select the parameter you want to display from the field opposite.
State descriptor of point at word	Applicable if the value is a state descriptor of a point in a user file, with the Point ID at the word specified in the field opposite. This option is generally only used by Honeywell personnel.
Parameter name of point at word	Applicable if the value is a parameter name of a point in a user file, with the Point ID at the word specified in the field opposite. This option is generally only used by Honeywell personnel.

.Checkbox



Property	Description
Label	The descriptive text that appears to the right of the checkbox.
Use inverse logic	Reverses the standard checkbox logic, so that it corresponds to the wording of **Label**. (In standard logic, checked equals “1”.) For example, labelling a checkbox “Open” would require the opposite logic to labelling it “Close”.

Combobox

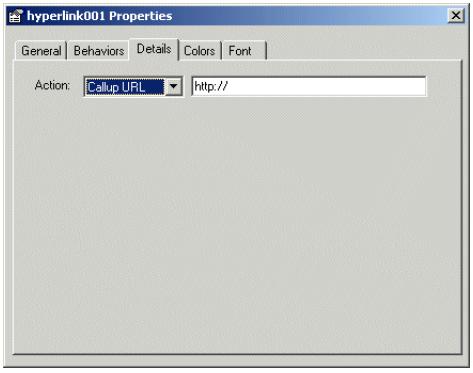


Property	Description
Display as Details	Specifies how the object is animated. See “Combobox display options” on page 184.
Sort alphabetically	Alphabetically sorts the items in the list.

Combobox display options

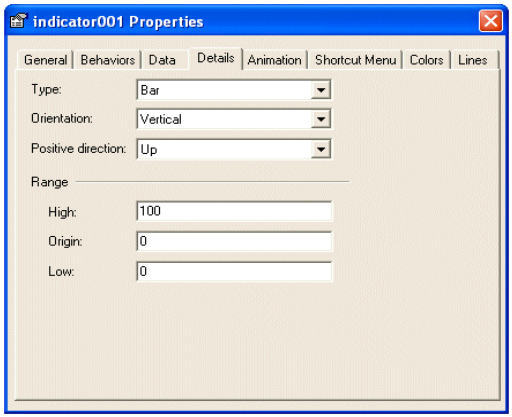
Display As option	Description
State Descriptor	Displays the defined states of the selected point as entries in the combobox’s list.
Acronym	Displays an acronym that corresponds to the (integer) value of the point parameter. See “Selecting acronyms” on page 219.
Parameter name of point at word	Only applicable if the value is a parameter name of a point in a user file, with the Point ID at the word specified in the field opposite. This option is generally only used by Honeywell personnel.

Hyperlink



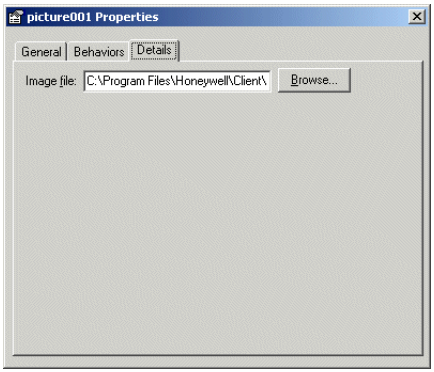
Property	Description
Action	Calls up the specified object: • `Callup URL`. Calls up the specified URL. See “Specifying URLs” on page 221. • `Callup Page`. Calls up the specified display (DSP or HMIWeb). Note that you do not have to specify the path because Station automatically searches its display folders. • `None`. Does nothing.

Indicator



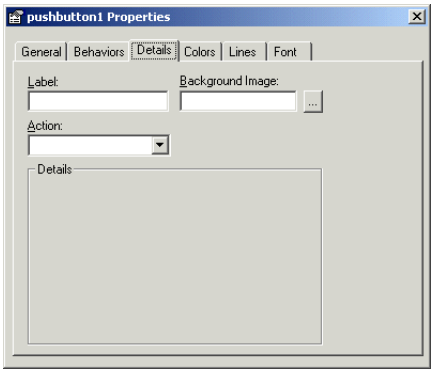
Property	Description
Type	Specifies how the object is animated: • `Bar`. A colored bar of variable height within the object indicates the relative value. • `Pointer`. A colored pointer moves up and down within the object to indicate the relative value.
Orientation	Specifies whether the indicator moves vertically or horizontally.
Positive direction	Specifies the fill direction of the indicator for a positive value. The options are Up and Down for vertical indicators and Left and Right for horizontal indicators.
High	Configures the upper bound of the indicator. This is a constant value that is a percentage of the actual range and can be greater than 100 and less than zero.
Origin	Configures the fill starting point of the indicator. This is a constant value that is a percentage of the actual range and can be greater than 100 and less than zero.
Low	Configures the lower bound of the indicator. This is a constant value that is a percentage of the actual range and can be greater than 100 and less than zero

Picture



Property	Description
Image file	The filename and path of the picture (graphic).

Pushbutton



Property	Description
Label	The pushbutton's label.
Background Image	The graphic that forms the pushbutton's background.
Action Details	The action performed when a user clicks the pushbutton. See "Defining a pushbutton's action" on page 188. Select None if you want to write a script for the button's onclick event.

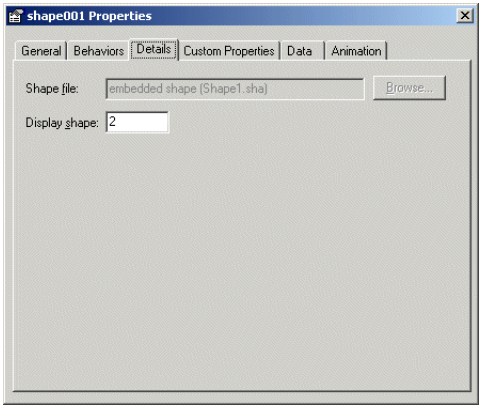
Defining a pushbutton's action

Select this action:	To do this:
None	Nothing. Select this if you want to write a script for the button's onclick event.
Callup Page	Call up the specified display or Web page. If you specify a Web page, you must ensure that it does not conflict with the Web access restrictions specified in Station. For details, see the *Configuration Guide*.
Current Display to	Send the current display to the specified Station within the console. You specify the Station in the Location box.
Request Report	Request the specified report.
Next Display to	Send the next display to the specified Station within the console. You specify the Station in the Location box.

Select this action:	To do this:
Request Task	Perform the task associated with the specified task LRN. In addition to specifying a task LRN, you must also specify: • Appropriate values for the task parameters • The security level required to perform the task For more information about the Server Display Program (LRN 21) actions and parameters, see the *Configuration Guide*.

Shape

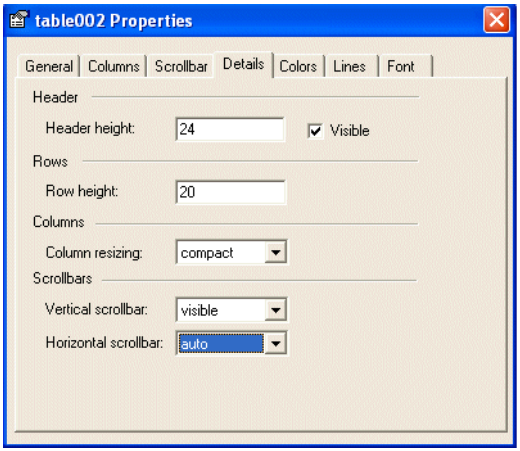
Applicable to both shape sequences and dynamic shapes, whether embedded or linked.



Property	Description
Shape file	The shape file on which this shape is based. Read-only.
Display shape	Only applicable to a shape sequence. The shape that is initially displayed when the display is called up. By default, this is the first shape in the sequence.

Table

To learn about tables, see “About tables” on page 241.



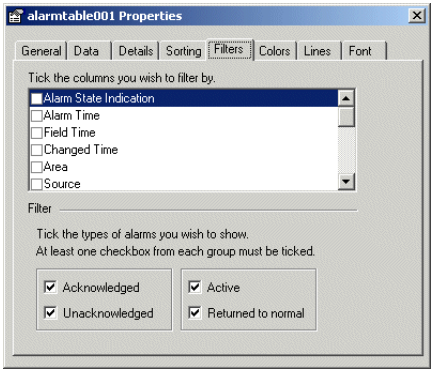
Property	Description
Header	
Header height Visible	**Header height** specifies the height (in pixels) of the column headings. If **Visible** is selected, the table includes column headings.
Rows	
Row height	The height (in pixels) of the rows.
Columns	
Column resizing	Specifies whether a user can resize columns, and the way in which adjacent columns are resized: • `None` = The columns cannot be resized • `Standard` = Resizing a column makes the column to the right wider/narrower • `Compact` = Resizing a column makes all columns to the right proportionally wider/narrower.

Property	Description
Scrollbars	
Vertical scrollbar	The vertical scrollbar's visibility setting. The values are: • `Hidden` = the scrollbar is hidden • `Visible` = the scrollbar is visible • `Auto` = the scrollbar only appears if the table cannot simultaneously show all the alarms.
Horizontal scrollbar	The horizontal scrollbar's visibility setting. The values are: • `Hidden` = the scrollbar is hidden • `Visible` = the scrollbar is visible • `Auto` = the scrollbar only appears if the table is narrower than total width of the columns.

Filters tab

The Filters tab is only applicable to alarm and event tables and specifies what types of alarms/events appear in the table.

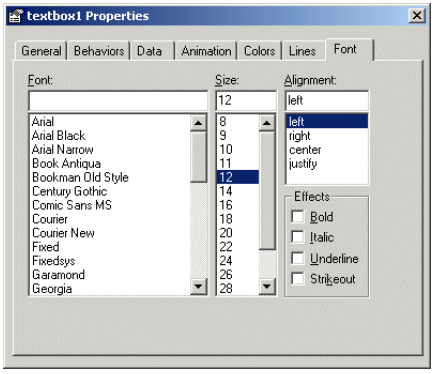
To learn about alarm and event tables, see “About alarm and event tables” on page 237.



Property	Description
Tick the columns you wish to filter by	Lists the alarm/event details (columns).
Filter	The properties that appear in this section depend on the currently selected column (alarm/event item). Configure the properties as appropriate. For example, if you only wanted to include unacknowledged alarms in the table, you would select the `Alarm State Indication` column, and then select **Unacknowledged** here. **Note:** You can filter by a column even if it does not appear in the table. (The Data tab specifies which columns are visible.) For example, you may want to filter out low and high priority alarms even though the `Priority` column is not visible.

Font tab

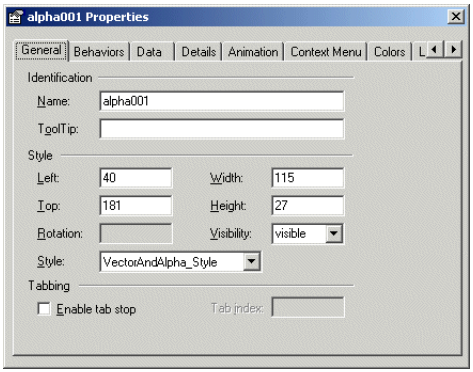
The Font tab controls the appearance of the object's text, such as the font and point size.



Property	Description
Font	Text is displayed in the selected font. Note that you should select fonts with care, and only use fonts that are suitable for computer displays.
Size	The text size, expressed in points. (If you want to specify a non-standard size, click the box and type the size—for example: **`56.5pt`**—and then press **ENTER**.)
Alignment	The horizontal alignment of text within the object.
Effects	The effect(s) applied to the text. For example, **Strikeout** draws a line through the text and **Underline** draws a line under it.

General tab

The General tab controls the object's basic properties such as its position and size.

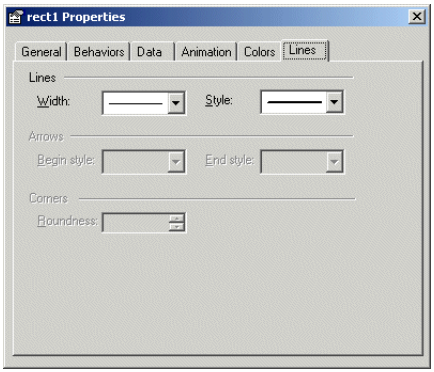


Property	Description
Name	The object's name. You use this name when referring to the object in scripts.
ToolTip	The text that appears when a user hovers the mouse pointer over the object. **Note:** Leave this blank if you select the **Hover** behavior.
Left Top Width Height	The object's position and size, expressed in pixels.
Rotation	Specifies the clockwise angle (degrees) through which the object is rotated. If grayed out, it indicates that the object cannot be rotated.
Visibility	The object's visibility setting. The values are: • `Hidden` = the object is always hidden • `Visible` = the object is always visible • `Inherit` = the object's visibility is determined by its parent (For example, if an object is part of a group, setting the visibility of the group will apply the same setting to the object.)

Property	Description
Style	The style applied to the object. The style is one of the styles defined in the style sheet attached to the display (Specified in **Stylesheet** on the display's Appearance tab.) To learn about style sheets, see "About style sheets" on page 108.
Enable tab stop	Allows users to "select" the object by pressing TAB. The integer number you type in **Tab index** specifies the order in which objects are selected. Note that: • Objects with a positive tab index are ordered before object(s) with a tab index of 0. • Objects with a negative tab index are excluded from the tabbing order. • Objects with the same tab index are ordered according to the order in which they appear in the HTML source for the display.

Lines tab

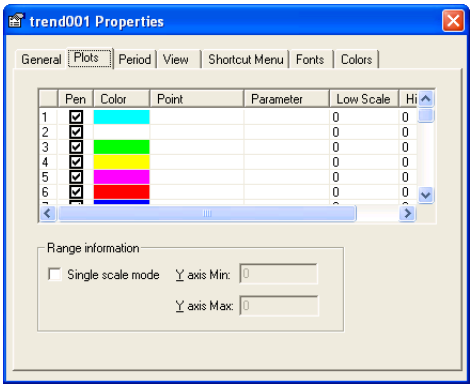
The Lines tab controls the object's line thickness and style.



Property	Description
Width	The width of the line.
Style	The line's style, such as dotted or dashed.
Arrows	Only applicable if the line has a start and an end. Specifies the type of "arrow" at the start and end of the line.
Roundness	Only applicable to a rounded rectangle. Increasing the value increases the "roundness" of the rectangle's corners.

Plots tab

The Plots tab specifies the properties of each plot in the trend.

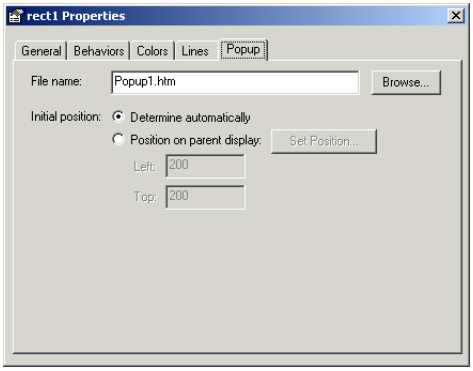


Property	Description
Pen	Select or clear the check box for each point whose plot you want visible on the trend.
Color	The pen color for the point parameter in the trend.
Point Parameter	The point parameter whose values you want to display. You can click the Point Browser button to browse points on this server or another server in the system. See “Point Browser” on page 36.
Low Scale	The minimum value visible on the trend for the point. This value is ignored if you select **Single scale mode**.
High Scale	The maximum value visible on the trend for the point. This value is ignored if you select **Single scale mode**.
Single scale mode	Select Single scale mode to use the same scale for all points in the trend.
Y axis Min	The minimum value shown on the y-axis.
Y axis Max	The maximum value shown on the y-axis.

Popup tab

The Popup tab specifies the properties of the popup that appears when a user clicks the object. The tab only appears if you select **Popup** on the Behaviors tab.

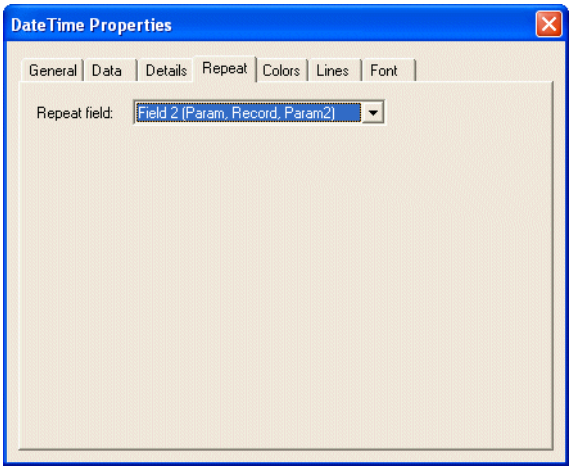
To learn about popups, see “About popups and faceplates” on page 59.



Property	Description
File Name of Page to be displayed in Popup Window	The filename of the popup display that appears when a user clicks the object.
Initial Position	Determines the initial position of the popup when it is called up. Select **Determine automatically** if you want Station to determine the initial position. Select **Position on parent display** if you want to specify the coordinates of the top-left corner of the popup. You can either type the **Left** and **Top** coordinates, or click **Set Position** and then click in the display to mark the position.

Repeat tab (table)

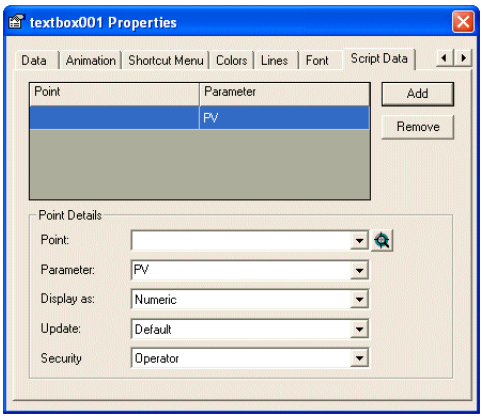
The Repeat tab lets you specify which field will be repeated in the table.



Property	Description
Repeat field	Specifies which field is repeated.

Script Data tab

The Script Data tab lists point parameters that can be accessed by scripts. (Note that this tab only appears if you select **Script Data** on the Behaviors tab.)



Notes

- To learn how add point parameters to this tab, see “Adding point parameters to an object” on page 203.
- To learn how to write scripts that access point parameter values, see “Reading and writing to point parameters” on page 304.
- You can add shape custom properties and system custom properties to this tab. (This enables scripts to access point parameters that have been mapped to the custom properties. See “Using custom properties in scripts” on page 309.)
- Adding a point parameter to one object makes it accessible to all scripts in the display. (However, it is generally more convenient to add point parameters and the scripts that access them to the same object.)

Property	Description
Point	The point ID and parameter name.
Parameter	If the object is in a dynamic shape, you can specify a custom property.
	You can click the Point Browser button to browse points on this server or another server in the system. See “Point Browser” on page 36.

Property	Description
Display as	The format used when reading (or writing to) the parameter, which can be: • `Numeric` • `State Descriptor` (Only applicable to a status point.) If you select `State Descriptor`, the value returned to a script will be a descriptor (such as "Stopped" or "Started"), rather than an integer (`0` or `1`).
Update	The frequency that the parameter value is updated: • `Default`. The value is updated at the Station update rate. • `On Demand`. The value is only updated when requested by a script using the RequestDemandData method. • `Once-Off`. The value is only updated when the display is initially called up or refreshed. • `1 second`, `2 second`, `4 second`, `5 second`, `10 second`, `30 second`, and `1 minute`. **Note:** The actual update rate is also determined by the Station and display update rates. See "Actual update rate" on page 202.
Security Level	The minimum security level required to allow scripts to write to the parameter. (The scripts inherit the security level of the current user.) Select `Read only` to prevent scripts from writing to the parameter, regardless of the current user's security level.

Actual update rate

If you specify a particular update rate, such as 5 seconds, the actual update rate may be different. The actual update rate is also controlled by the Station update rate (specified when configuring Station) and the **Display refresh rate** (specified on the display's Details tab).

The server selects the slowest rate, and rounds this down to a multiple of the slower of the Station and display rates. The following table shows examples of actual update rates for common combinations (all values are in seconds).

Station update rate	Display update rate	Parameter update rate	Actual update rate
1	1	1	1
1	5	1	5
5	1	1	5

Station update rate	Display update rate	Parameter update rate	Actual update rate
1	1	5	5
5	1	9	5
5	1	10	10
5	1	14	10
5	1	15	15

Adding point parameters to an object

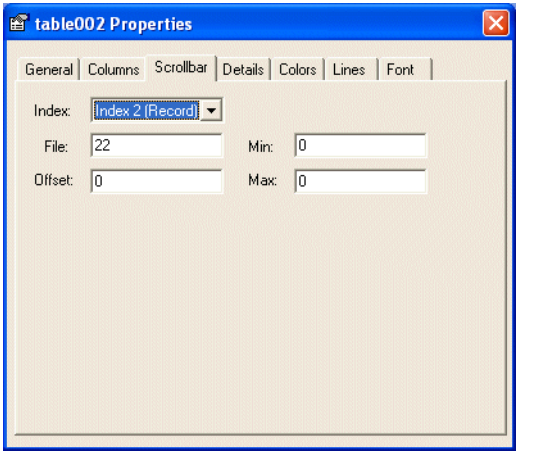
To add one or more point parameters to an object:

1. Select the object.
2. Click the Behaviors tab tab on the Properties Window and select **Script Data**. This adds the Script Data tab to the Properties Window.
3. Click the Script Data tab.
4. Click **Add** to add a blank row.
5. Specify the point details.
6. Repeat steps 4 and 5 for any other point parameters you want to add.

Scrollbar tab

The Scrollbar tab specifies properties for the scrollbar that can control:

- A value in the server database. For example, you could use a scrollbar to control the value of a setpoint.
- An offset to a record or field for the display. Used in this way, the scrollbar controls any other display objects that use the offset.



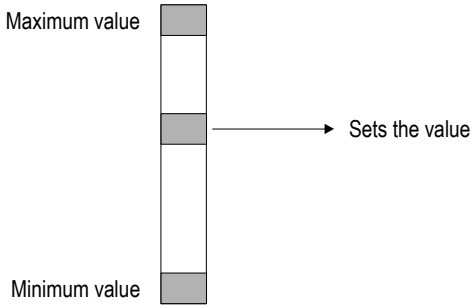
Property	Description
Index	Specifies which index is used to display data in the table.
File	Specifies the file whose record, field or word is being offset.
Min	The minimum value to which the scrollbar can scroll. If you select `Point/Parameter` or `Database File` from **Type of database link** on the Data tab, this property is ignored and the minimum database value is used.
Offset	Controls the current record, field or word offset for the display. When a user moves the scroll box, the current offset changes.

Property	Description
Max	The maximum value that the scrollbar can scroll to. If you select `Point/Parameter` or `Database File` from **Type of database link** on the Data tab, this property is ignored and the maximum database value is used.

Using a scrollbar to set a value

The following figure shows how to use a scrollbar to set a value in the server database. When a user moves the scroll box up or down, the value varies accordingly between the minimum and maximum values.

Figure 20 Setting the value with a scrollbar

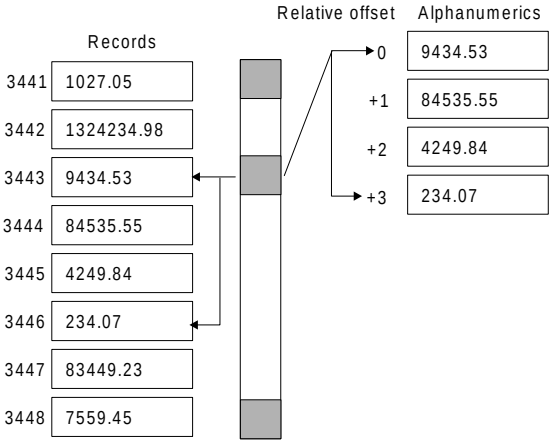


Using a scrollbar to control the offset

The following figure shows how to use a scrollbar to display the contents of records in a large file. The alphanumerics are configured so they display five adjacent records.

When a user moves the scroll box up or down, the display offset changes and the contents of the alphanumerics change accordingly. In this example, the alphanumerics display the values of records 3443 to 3446.

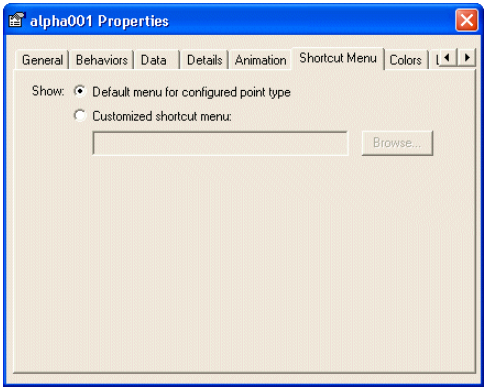
Figure 21 Controlling the offset with a scrollbar



Shortcut Menu tab

The Shortcut Menu tab specifies the object's shortcut menu, which appears when a user right-clicks the object.

To learn about shortcut menus, see "About shortcut menus" on page 125.

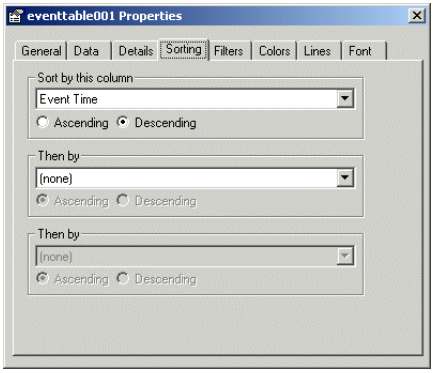


Property	Description
Show	Specifies which shortcut menu is used: • **Default menu for configured point type**. Uses the default menu for the associated point. For example, if the object is linked to an analog point, the standard shortcut menu for an analog point will be used. (The display's default menu is used if the object is not linked to a point/parameter.) • **Customized shortcut menu**. Uses the specified custom menu.

Sorting tab

The Sorting tab is only applicable to alarm and event tables and the table object. It specifies how alarms/events are sorted.

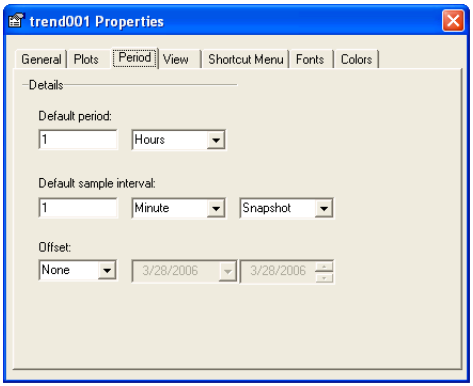
To learn about alarm and even tables, see “About alarm and event tables” on page 237.



Property	Description
Sort by this column Then by	These properties specify up to three levels of sorting instructions for alarms/events that appear in the table. For example, you might want to sort alarms by `Priority` (select **Descending** to list the highest priority alarms first) and then by `Date & Time` (select **Descending** to list the newest alarm first). At each level, select a column (alarm/event item) and the order in which alarms/events are sorted. **Note:** You can sort by a column even if it does not appear in the table. (The Data tab specifies which columns are visible.) For example, you may want the most recent alarm to appear at the top of the table even though the `Alarm Time` column is not visible.

Period tab

The Period tab controls the time properties of a trend.



Property	Description
Default period	The amount of time for which you want to see data. Specify the time units and the number of units.
Default sample interval	The history interval. You can choose to see snapshots or averages. Choose one of the sample periods supported by the server: • 1 second snapshot • 1 minute snapshot • 6 minute average • 1 hour average • 8 hour average • 24 hour average • 1 hour snapshot • 8 hour snapshot • 24 hour snapshot

Property	Description
Offset	The offset works in conjunction with the default period. By specifying an offset, you can choose which period is shown in the trend. If you select **Left**, specify the starting date and time. If you select **Center**, specify the date and time around which the default period is centered. If you select **Right**, specify the end date and time.

Notes

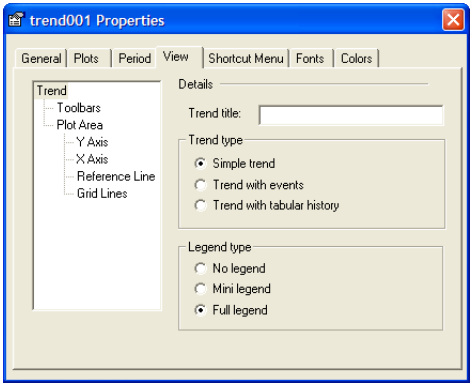
- The number of samples in your trend (calculated using the default period and the default sample interval) should be within 10 and 4096 samples. If the number of samples falls outside of this range, the period is automatically adjusted at runtime so that the sample limit is not exceeded. For example, if you specify a default time period of three hours with a 1 second snapshot sample interval, the number of samples would be 10800. When the display is called up the period would be changed from three hours to the next closest period that gives a sample of less than 4096. In this example, a period of 1 hour would be used.
- The same limits apply to the runtime operation of a trend, however at runtime, both the period or the sample interval can be automatically adjusted. For example, an operator selecting a new period can result in the sample interval automatically adjusting; likewise selecting a new sample interval can result is the period automatically adjusting.

View tab

The View tab contains properties that are specific to each component of a trend.

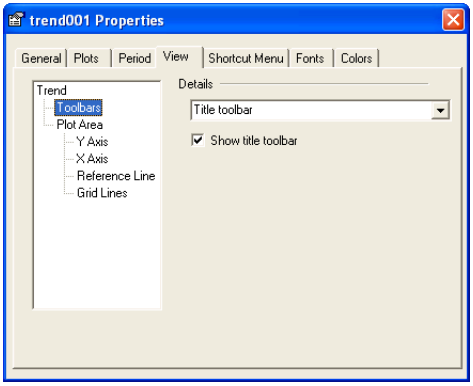
For details about this type of object:	Go to:
Trend	page 212
Toolbars	page 213
Plot Area	page 214
Y Axis	page 215
X Axis	page 216
Reference Line	page 217
Grid Lines	page 218

Trend



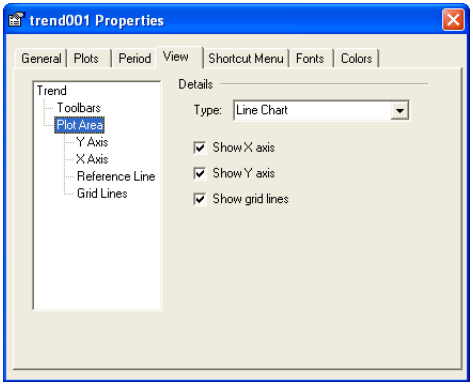
Property	Description
Trend title	The title of this trend. The title of the trend is visible only if you are also showing the Title toolbar.
Trend type	The type of trend: • Simple trend. The trend shows plotted data for the specified point parameters. • Trend with events. The trend shows plotted data for the specified point parameters as well as an Event Summary. • Trend with tabular history. The trend shows plotted data for the specified point parameters as well as tabular history for the specified point parameters.
Legend type	The type of legend included with the trend: • None. No legend is included. • Mini legend. The legend contains the point ID and parameter and a check box to show or hide the plot. • Full legend. The legend contains the point ID, parameter, description, low and high scale, current value, reference vale, engineering units, and a check box to show or hide the plot.

Toolbars



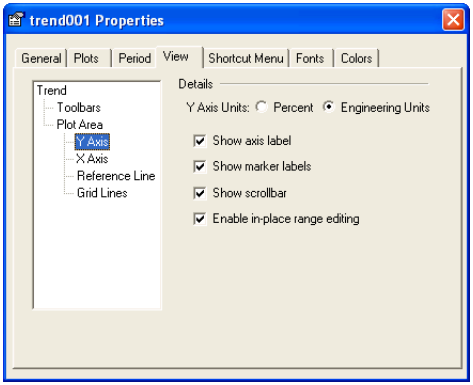
Property	Description
Details	You show or hide the following toolbars: • Title toolbar • Chart toolbar • X axis toolbar Select the toolbar from the list and select or clear the **Show toolbar** check box to show or hide the toolbar.

Plot Area



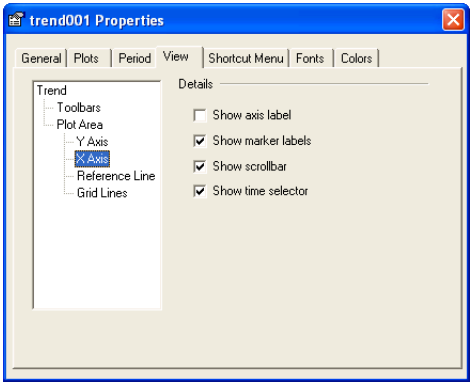
Property	Description
Type	The plot type: • Line chart • Bar chart You can view bar chart for only one point at a time. By default this is the first point on the trend.
Show plot selector	Shows or hides the plot selector. The plot selector provides a way of changing the highlighted plot in a line chart or the point in a bar chart.
Show X axis	Shows or hides the x-axis. If you hide the x-axis, the x-axis toolbar is also hidden, even if you have selected to show the x-axis toolbar.
Show Y axis	Shows or hides the y-axis.
Show grid lines	Shows or hides the gridlines.

Y Axis



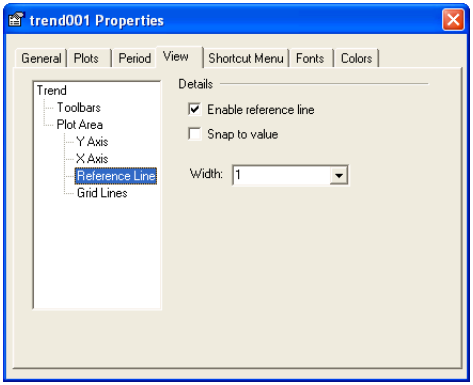
Property	Description
Y axis units	The type of units shown on the y-axis: • Percent • Engineering units
Show axis label	If selected, the labels on the axis are visible.
Show marker labels	If selected, the scale marker labels on the axis are visible.
Show scrollbar	If selected, the scrollbar on the axis is visible.
Enable in-place range editing	If selected, a user can edit the visible range of a point using controls on the axis.

X Axis



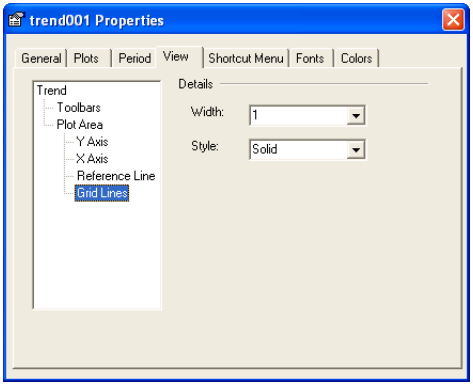
Property	Description
Show axis label	If selected, the axis label if visible.
Show marker labels	If selected, the scale marker labels on the axis are visible.
Show scrollbar	If selected, the scrollbar on the axis is visible.
Show time selector	If selected, the time control on the axis is visible.

Reference Line



Property	Description
Enable reference line	If selected, a user can add a reference line to the trend.
Snap to value	Causes the reference line to be placed on the nearest value.
Width	The width, in pixels, of the reference line.

Grid Lines



Property	Description
Width	The width, in pixels, of the grid lines.
Style	The style of the grid lines: • Solid • Dashed • Dotted • Dash dot • Dash dot dot

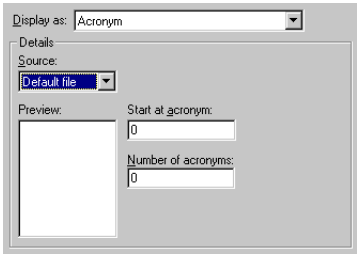
Selecting acronyms

Related acronyms are stored as contiguous records in the order corresponding to the parameter values. To select a set of acronyms, you specify the record of the first acronym, as well as the number of acronyms required.

To select default acronyms:

1 Select `Acronym` in **Display as** and `Default file` in **Source**.

2 Specify the values of the other properties as appropriate.

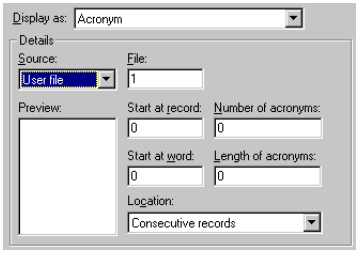


Property	Description
Start at acronym	The record that contains the first acronym. System acronyms are stored in records 1 to 2048, and user-defined acronyms are stored in records 2881 to 3880.
Number of acronyms	The number of acronyms you want to use.
Preview	If you are on the server, the selected acronyms appear in this box.

To select custom acronyms:

1 Select `Acronym` in **Display as** and `User file` in **Source**.

2 Specify the values of the other properties as appropriate.



Property	Description
File	The file that contains the custom acronyms.
Start at record	The record that contains the first acronym.
Number of acronyms	The number of acronyms you want to use.
Start at word	The word that contains the first acronym.
Length of acronyms	The length of the acronyms.
Location	Specifies how the acronyms are stored within the file: • `Consecutive records` • `Single record`, where the acronyms are stored in consecutive words within the same record.
Preview	If you are on the server, the selected acronyms appear in this box.

Specifying URLs

The general format for the *URL* (Uniform Resource Locator) of a file or Web site is:

Protocol://*ResourceName*

Part	Description
Protocol	The protocol, which can be `http`, `ftp`, `gopher`, or `file`. The default protocol is `file`. If the protocol is not specified, Station assumes that the URL refers to a display, either on the Station computer or on the network.
ResourceName	The resource address or identifier, such as a file or Web address. If the extension is not specified, Station assumes that it is a display file.

Examples

The URL for a HTML file called “procedure.htm” can be specified as either:

```
file://c:\procs\procedure.htm
```

or:

```
c:\procs\procedure.htm
```

The URL for a Web address:

```
http://www.ourwebsite.com
```

The URL for a Word file called “sp.doc”:

```
file://c:\procs\sp.doc
```

About vector graphic objects

Vector graphic objects are “line-based” objects such as rectangles and curves. The following objects are vector graphic objects.

- Arc
- Bezier Curve
- Line
- Oval
- Polygon
- PolyLine
- Rectangle
- Rounded Rectangle
- Textbox
- Wedge

The following table lists specialized editing tasks you can perform on vector graphic objects.

Task:	Go to:
Create a new object from overlapping objects.	page 72
Combine objects to improve the display’s performance.	page 83
Change an object’s shape with the node editor.	page 88

About trends

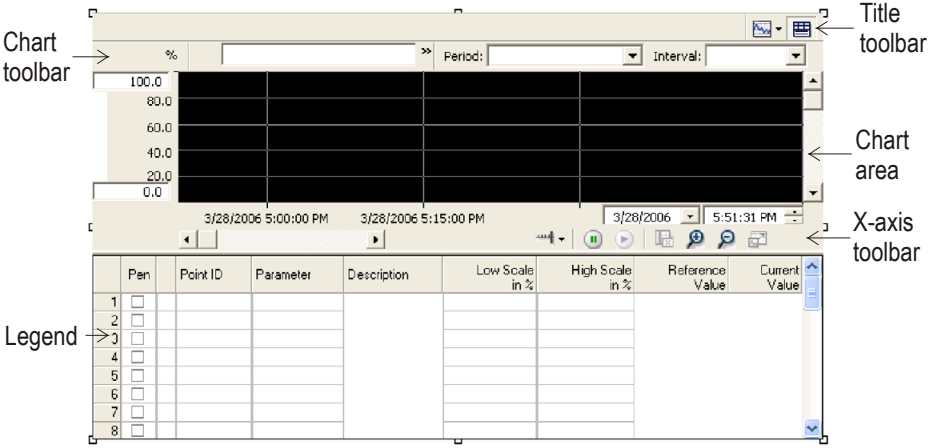
A trend object presents information in a graphical manner, and can be configured in numerous ways.

Notes

- Some of the features described here can only be accessed using scripts. The Properties Window only allows you to configure a chart's basic properties.
- Currently, you can only create line and bar trends in HMIWeb Display Builder. If you want to create another type of chart, such as pie or tuning, you must use Display Builder.

A trend includes the following main components:

- Chart area. The data display area.
- Axis. The rulers and legends of the x/y axes.
- Toolbar. The toolbar above the chart area.
- Legend. The tabular area below the chart area.



If the size of the Station window or the zoom level is too small to show the entire trend, some components are automatically hidden, such as the axes and toolbars, so that the chart area is always visible. The components are visible when the Station window or zoom level is big enough to display them.

If you know you only have a limited area to display the trend, consider whether you can exclude these components when you create your custom trend.

Chart area

A trend can display up to 32 plots, each of which can be separately configured.

You can also configure the view's appearance such as the background and grid colors.

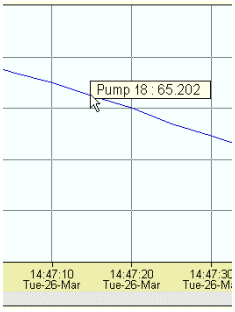
Samples and sample intervals

For the x-axis, the minimum sample interval is 1 second. (The maximum sample interval is 24 hours.)

Plot ToolTips

When a user moves the pointer over a plot, a ToolTip shows the value of the plot at that point, as shown in the following figures.

With scripting, you can include a suitable prefix to the value, as shown in the figure on the right.

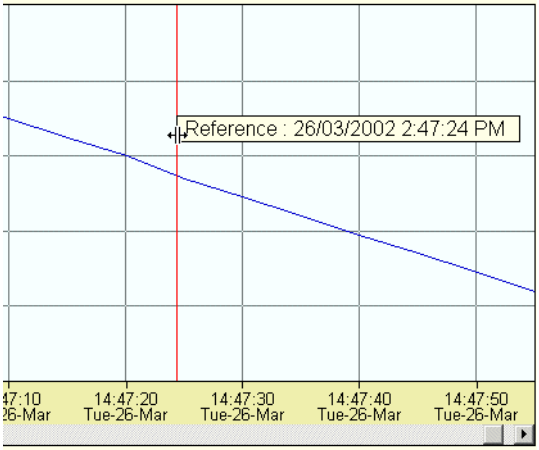


Reference line and reference values

When a user clicks in the Plot View a vertical *reference line* appears. The placement of the reference line depends on whether snap to value is set. If snap to value is set, the reference line is place to nearest value. If snap to value is not set, the reference line passes through the point marked by the click. which passes through the point marked by the click. Also, when the user moves the pointer over the reference line, a ToolTip appears which shows its date/time.

With scripting, you use the OnReferenceCursorSet to obtain the *reference values*. A reference value is the value (and date/time) of a plot where it intersects the reference line. (The value is calculated using linear interpolation between the two adjacent samples.)

If the Legend is visible, the reference values are shown in the Legend grid.

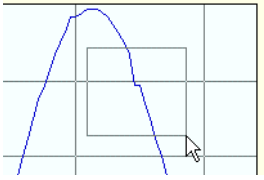


Event bar and data tips

If the Event Summary is visible with the trend, an event bar is added to the bottom of the chart area. The event bar contains icons to indicate when an event occurred. A data tip appears when a user clicks on an event symbol. To remove the data tip a user clicks on the event symbol again. If a different symbol is clicked, the data tip displayed is for the new event symbol. Only one data tip can be visible at a time. If multiple events occurred in the region of the event symbol then arrow buttons appear in the data tip. These arrow buttons allow users to scroll through the events that occurred at that time.

Zoom function

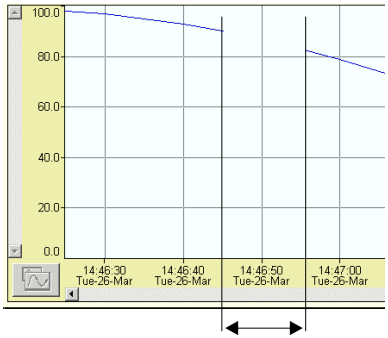
A user can zoom into a particular part of the trend by dragging diagonally across the area of interest, as shown in the following figure. When the user releases the mouse button, the trend zooms into the rectangle marked by the pointer.



The user returns the trend to its normal scale by clicking the button.

Invalid data

If the trend receives invalid data, it discontinues the plot until valid data is again received, as shown in the following figure.

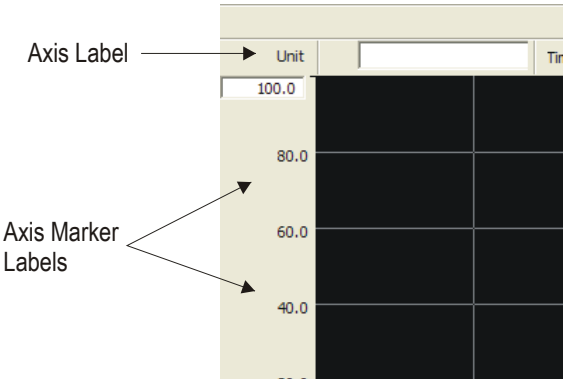


Invalid Data Region

Axis

By default, the Axis View includes the axis marker labels.

With scripting, you can also include axis labels.



Labels with multiple plots

Each plot on a trend can have its own axis and marker labels (as well as other properties such as range and number of samples). However, only the labels for the first plot in a trend are displayed. The ranges of the other plots are automatically scaled to fit the y-axis.

Toolbar

A trend can include toolbars so that users can manipulate the trend. With the toolbar, users can:

- Change the scale
- Change the line plot to a bar graph for a single point
- Show or hide the Event Summary
- Show or hide the Legend
- Show or hide tabular history
- Zoom
- Change the period

With scripting, you use the SetXAxisToolbarItemVisible method to specify what items are available on the toolbar.

Legend

The Legend provides more detailed information about the points that are included in the trend.

The legend can be provided in two forms, a full legend or a mini legend. The mini legend contains the pen check box, point ID, and parameter. In addition to these items, the full legend also contains the point description, low scale, high scale, current value, reference value and engineering units.

If the toolbar is visible on the trend, users have the option to show or hide the legend.

If the trend also includes events or includes tabular history, then only the mini legend can be included.

Scripting trends

One of the features of the trend is the ability to write scripts that can be used to manipulate the user interface and plot of the trend.

User interface manipulation

There are a number of simple ways that you can customize the look of a trend such as hiding or showing the legend and gridlines, enabling or disabling zooming, and changing the title and color of the trend canvas. You can also use scripts to add a point to the trend or change the trend view based on user input.

Adding a Point

The following script can be added to a button to add a Sinewave point to a trend:

```
Sub pushbutton001_onclick
customtrend001.Legend.Parameter(1) = "pv"
customtrend001.Legend.PointID(1) = "Sinewave"
End Sub
```

Changing trend views

The following script can be added to a button object to change the trend view based on input from the user:

```
Sub pushbutton002_onclick
dim TView
TView = Inputbox ("Enter View mode, Events, Numeric, Single")
if TView = "Events" then
customtrend001.view = 2
msgbox "The Trend with Events view has been activated"
else
if Tview = "Numeric" then
customtrend001.view = 1
msgbox "The Trend with Tabular History view has been activated"
else
if TView = "Single" then
customtrend001.view = 0
msgbox "The Single Trend view has been activated"
else
msgbox "You have entered an invalid value. Please use x, y, and
   xy only"
end if
end if
end if
```

Toggling the legend

The following script can be attached to a pushbutton object to hide and show the trend's legend:

```
Sub pushbutton005_onclick
customtrend001.Legend.Visible = not
   customtrend001.Legend.Visible
End Sub
```

Toggling the toolbar

The following script can be attached to a pushbutton object to hide and show the toolbar:

```
Sub pushbutton003_onclick
customtrend001.ChartToolbar.Visible = not
   customtrend001.ChartToolbar.Visible
End Sub
```

Toggling the zoom

The following script can be attached to a pushbutton object to enable and disable the zoom function for the trend:

```
Sub pushbutton009_onclick
customtrend001.ChartArea.PlotCanvas.EnableZooming = not
   customtrend001.ChartArea.PlotCanvas.EnableZooming
if customtrend001.ChartArea.PlotCanvas.EnableZooming = 0 then
msgbox "Zooming has been disabled"
else
msgbox "Zooming has been enabled"
end if
End Sub
```

Toggling the gridlines

The following script can be attached to a pushbutton object to hide and show the gridlines on the trend canvas:

```
Sub pushbutton012_onclick
customtrend001.ChartArea.PlotCanvas.GridLinesVisible = not
   customtrend001.ChartArea.PlotCanvas.GridLinesVisible
End Sub
```

Changing the canvas color

The following script can be attached to a pushbutton object to change the color of the canvas. This example will change the canvas color to green:

```
Sub pushbutton011_onclick
customtrend001.ChartArea.PlotCanvas.BackgroundColor = vbGreen
End Sub
```

Changing the trend title

The following script can be attached to a pushbutton object to change the title of the trend:

```
Sub pushbutton008_onclick
dim TTitle
TTitle = InputBox("Enter Trend Title")
customtrend001.TitleToolbar.TitleText = TTitle
End Sub
```

Plot manipulation

There are many different ways you can customize the look of trends using scripts. Several examples are listed below.

Changing the plot interval

You can add a pushbutton to a display that will change the trend to the interval defined in the script. You cannot have custom intervals (only custom periods) so if you set an interval in a script, it has to be one of the values in the drop-down menu on the trend toolbar, in the format hours, minutes, seconds, milliseconds. The following example will set the trend to a six minute interval:

```
Sub pushbutton006_onclick
customtrend001.SetSampleInterval 0,6,0,0, true
End Sub
```

Changing the period

You can create a script to change the trend's period to a value specified by the user. In the following script example, the user is only allowed to enter a value for hours, minutes and seconds, but this can be expanded to days, week, months, and years.

```
Sub pushbutton007_onclick
dim hours
hours = Inputbox ("Enter the time period in hours")
dim minutes
minutes = Inputbox ("Enter the time period in minutes")
dim seconds
seconds = Inputbox ("Enter the time period in seconds")
customtrend001.SetPeriod 0,0,0,0,hours,minutes,seconds '
   customtrend001.SetPeriod years, months, weeks, days, hours,
   minutes, seconds
sgbox "You have set the time period to be " + hours + " hours "
   + minutes + " minutes " + seconds + " seconds "
End Sub
```

Changing the period

You can use a script to change the trend to show historical data from a period that is entered by the user. This is useful for quickly comparing historical data from a set point in time, such as a benchmark that can be hard coded into the script. To set the date and time to which the trend will be set, it must be in the format of 9:00:00AM May 5, 2005.

```
Sub pushbutton002_onclick
Chandim userleftdate
userleftdate = Inputbox ("Enter the first time period of the
   trend you would like to view, eg 9:00:00 AM May 5 2005")
dim userrightdate
userrightdate = Inputbox ("Enter the second time period of the
   trend you would like to view, eg 10:00:00 AM May 5 2005")
dim leftDate, rightDate
leftDate = CDate(userleftdate)
rightDate = CDate(userrightdate)
customtrend001.SetTimeRange leftDate, 0, rightDate
End Sub
```

Changing the plot color

You can change the color of the plot using a script. The following script will change the plot to blue.

```
Sub pushbutton010_onclick
if customtrend001.Legend.PointID(1) <> "Sinewave" then
msgbox "Please click on Add Point first"
else
customtrend001.Legend.PlotColor(1) = VBblue
end if
End Sub
```

Changing the plot label

Another use for scripts on a trend is to allow you to change the labels for the plot as well as the X and Y axes.

```
Sub pushbutton013_onclick
dim PLabel
PLabel = InputBox("Enter Plot Label")
dim xLabel
xLabel = InputBox("Enter X-axis Label")
dim yLabel
yLabel = InputBox("Enter Y-axis Label")
customtrend001.ChartArea.XAxis.LabelVisible = True
customtrend001.ChartArea.YAxis.LabelVisible = True
customtrend001.ChartArea.Plots.Item(1).SetLabels PLabel, xLabel,
   yLabel
End Sub
```

Changing the zoom mode

Zoom mode allows you to zoom in on the X, Y or both axes. You can create a script that allows the user to select which of the axes they want to zoom in on.

```
Sub pushbutton003_onclick
dim ZMode
ZMode = Inputbox ("Enter Zoom mode, x, y, or xy")
if ZMode = "x" then
customtrend001.ChartArea.PlotCanvas.ZoomMode = 1
msgbox "The Trend will now Zoom to the X-Axis only"
else
if ZMode = "y" then
customtrend001.ChartArea.PlotCanvas.ZoomMode = 2
msgbox "The Trend will now Zoom to the Y-Axis only"
else
if ZMode = "xy" then
customtrend001.ChartArea.PlotCanvas.ZoomMode = 0
msgbox "The Trend will now Zoom to both axes"
else
msgbox "You have entered an invalid value. Please use x, y, and
   xy only"
end if
end if
end if
End Sub
```

Changing the visible range

Visible range allows you to change the Y axis of the trend to reflect only a range of values. Note that this function is only valid on non single scale plots so after entering valid values, make sure you disable single scale for all plots by clicking the second button on the top left of the trend on the toolbar.

```
Sub pushbutton004_onclick
if customtrend001.Legend.PointID(1) <> "Sinewave" then
msgbox "Please click on Add Point first"
else
dim yLow
yLow = InputBox ("Enter Y-Axis Low Value")
dim yHigh
yHigh = InputBox ("Enter Y-Axis High Value")
if yLow > yHigh then
msgbox "The low value you have entered is larger than the high
   value. Please re-enter your values"
else
customtrend001.ChartArea.Plots(1).SetYAxisScale yLow, yHigh
end if
end if
End Sub
```

Changing the chart type

You can write a script that will allow the user to toggle between line and bar charts.

```
Sub pushbutton014_onclick
if customtrend001.ChartArea.ChartType = 1 then
customtrend001.ChartArea.ChartType = 2
msgbox "Line chart view activated"
else
customtrend001.ChartArea.ChartType = 1
msgbox "Bar chart view activated"
end if
End Sub
```

Changing the marker style

A script can also be used to modify the marker styles used on a plot based on user input. Note that there are more marker styles available than displayed in the following script. For more information on marker styles see “SetPlotMarkerStyle method” on page 541.

```
Sub pushbutton005_onclick
if customtrend001.Legend.PointID(1) <> "Sinewave" then
msgbox "Please click on Add Point first"
else
if customtrend001.ChartArea.ChartType = 1 then
msgbox "Please change the Chart Type back to line by clicking
   the Chart Type button"
else
dim MStyle
MStyle = Inputbox ("Enter a marker style, None , Cross, Circle
   or Square")
if MStyle = "None" then
customtrend001.ChartArea.Plots(1).MarkerStyle = 0
msgbox "No marker style set"
else
if MStyle = "Cross" then
customtrend001.ChartArea.Plots(1).MarkerStyle = 1
msgbox "Cross marker style set"
else
if MStyle = "Square" then
customtrend001.ChartArea.Plots(1).MarkerStyle = 3
msgbox "Square marker style set"
else
if MStyle = "Circle" then
customtrend001.ChartArea.Plots(1).MarkerStyle = 7
msgbox "Circle marker style set"
else
msgbox "You have entered an invalid value. Please use None,
   Cross, Square, Circle only"
end if
```

```
end if
end if
end if
end if
end if
End Sub
```

Changing the line style

A script can also be used to modify the plot line styles based on user input. Note that there are more line styles available than displayed in the following script. For more information on marker styles see “SetLineStyle method” on page 535.

```
Sub pushbutton008_onclick
if customtrend001.Legend.PointID(1) <> "Sinewave" then
msgbox "Please click on Add Point first"
else
if customtrend001.ChartArea.ChartType = 1 then
msgbox "Please change the Chart Type back to line by clicking
   the Chart Type button"
else
dim LStyle
LStyle = Inputbox ("Enter a line style, Solid, Dash, Dot or
   Dashdot")LWidth = Inputbox ("Enter a thickness for the plot
   line, eg 5")
if LStyle = "Solid" then
customtrend001.ChartArea.Plots(1).SetPen 0, LWidth, vbred
msgbox "Solid line style set"
else
if LStyle = "Dash" then
customtrend001.ChartArea.Plots(1).SetPen 1, LWidth, vbred
msgbox "Dash line style set"
else
if LStyle = "Dot" then
customtrend001.ChartArea.Plots(1).SetPen 2, LWidth, vbred
msgbox "Dot line style set"
else
if LStyle = "Dashdot" then
customtrend001.ChartArea.Plots(1).SetPen 3, LWidth, vbred
msgbox "Dashdot line style set"
else
msgbox "You have entered an invalid value. Please use Solid,
   Dash, Dot, or Dashdot"
end if
end if
end if
end if
end if
end if
End Sub
```

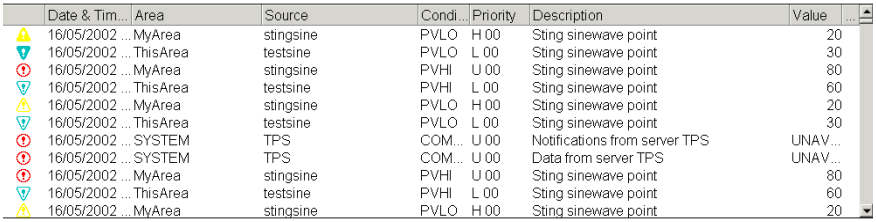
About alarm and event tables

An *alarm table* lists alarms in a manner similar to the Alarm Summary. An *event table* lists events in a manner similar to the Event Summary.

The properties you can specify when creating an alarm or event table include:

- The types of alarms/events that are shown (For example, you may only want to show urgent alarms for a particular area.)
- The details that are shown (date/time, description, priority and so on), and the order in which they appear across the table
- Whether there are horizontal and vertical scrollbars

Figure 22 Typical alarm table

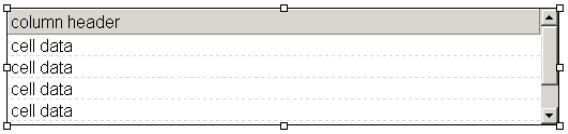


	Date & Tim...	Area	Source	Condi...	Priority	Description	Value
	16/05/2002 ...	MyArea	stingsine	PVLO	H 00	Sting sinewave point	20
	16/05/2002 ...	ThisArea	testsine	PVLO	L 00	Sting sinewave point	30
	16/05/2002 ...	MyArea	stingsine	PVHI	U 00	Sting sinewave point	80
	16/05/2002 ...	ThisArea	testsine	PVHI	L 00	Sting sinewave point	60
	16/05/2002 ...	MyArea	stingsine	PVLO	H 00	Sting sinewave point	20
	16/05/2002 ...	ThisArea	testsine	PVLO	L 00	Sting sinewave point	30
	16/05/2002 ...	SYSTEM	TPS	COM...	U 00	Notifications from server TPS	UNAV...
	16/05/2002 ...	SYSTEM	TPS	COM...	U 00	Data from server TPS	UNAV...
	16/05/2002 ...	MyArea	stingsine	PVHI	U 00	Sting sinewave point	80
	16/05/2002 ...	ThisArea	testsine	PVHI	L 00	Sting sinewave point	60
	16/05/2002 ...	MyArea	stingsine	PVLO	H 00	Sting sinewave point	20

Notes

- Unlike an alarm table, an event table does not show events as soon as they occur (subject to the display's refresh rate). Instead, it shows events stored in the *SQL events database*, which is updated from the "live" *system database* at 30-second intervals. This means that the most recent event in an event table will be at least 30 seconds old. For more details about the two databases, see the *Configuration Guide*.
- By default, the event table has its event time filter set to "TODAY", which means only those events that are generated on the same day are displayed. The "TODAY" event time filter shows events stored in the SQL events database, which is updated from the "live" system database at 30-second intervals.
- When you create an alarm or event table in HMIWeb Display Builder, its appearance does not closely reflect its appearance in Station. For example, there is only one "column" regardless of how many columns you actually specify.

Figure 23 Typical alarm or event table as it appears in HMIWeb Display Builder



Setting alarm states

When you select the **Show point disabled icon** option on the alarmstate Properties dialog box the alarm icon would show the point disabled icon D if the point linked to the icon has no alarm and has been disabled. The alarm state object needs to be bound to an enable/disable state parameter, for example AlarmDisabled for a SCADA point.

This option is not selected by default. When the option is selected for an alarm state object, the object will not show the point disabled icon, you have to perform the following steps:

1 In the Properties window of an alarm state object, click the Details tab and then select the **Show point disabled icon** option.

2 Click the Data tab and enter the point link with this object. For the parameter setting, you have to select one of the following parameters depending on the point type:

Point type	Paramater to use
TPS	ALENBST
C200	ALMENBSTATE
SCADA	ALARMDISABLED

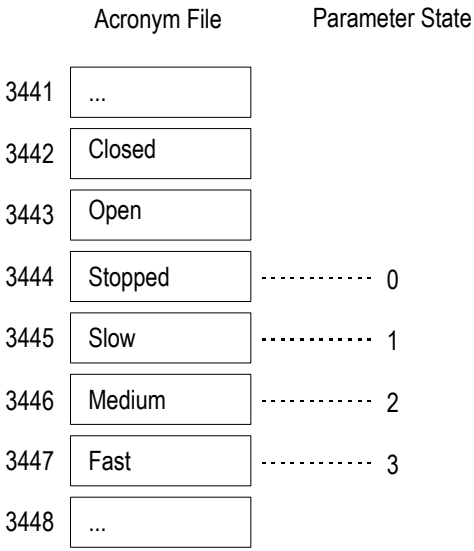
About acronyms

An *acronym* describes the meaning of a parameter's state (or integer value). For example, the acronyms "Stopped" and "Running" are much more meaningful in displays than the raw parameter values "0" and "1".

The *default acronym file* contains acronyms suitable for most purposes. Acronyms in this file are divided into two ranges:

- *System*. Pre-defined, which include commonly required acronyms. System acronyms are stored in records 1 to 2048.
- *User-defined*. Stored in records 2881 to 3880. You define user acronyms in Station.

Related acronyms are stored as contiguous records in the order that corresponds to the states of a parameter. The following example shows how the four states of a parameter correspond to a set of four acronyms—`Stopped` to `Fast`. (When selecting a set of acronyms, you specify the record of the first acronym and the number of acronyms required—see "Selecting acronyms" on page 219.)



Custom acronym files

If necessary, you can create a *custom acronym file*—the main advantage is that it allows you to define acronyms that are more than 10 characters long.

For details about creating custom acronyms, see the *Display Building Guide*.

About tables

A *table* lists data in a tabular format. The data can include alphanumeric, checkbox, pushbutton and combobox objects.

The properties you can specify when creating a table include:

- Whether the header row is visible
- Whether the table is resizeable
- Whether there are horizontal and vertical scrollbars

Figure 24 Example table

Trend	Description	Type	Sample Interval
1	Tank 1 and 2 Flow	Multiplot	5 second
2	Tank 3 and 4 Flow	Multiplot	5 second
1	Tank 1 and 2 Flow	Multiplot	5 second
2	Tank 3 and 4 Flow	Multiplot	5 second
1	Tank 1 and 2 Flow	Multiplot	5 second
2	Tank 3 and 4 Flow	Multiplot	5 second

New Trend Delete Trend

Creating a table

To create a table:

1 Click on the Table icon on the Builder toolbar

2 Draw the table on the display.

3 Use the Properties dialog box to add columns to the table and modify the properties and layout of the table.

To add columns to the table:

1 Display the Columns tab on the table's Properties dialog window.

2 Click on the Add New button.

The Column Properties dialog window appears.

3 Type the name of the column into the Name field. The column name is used internally within HMIWeb Display Builder.

4 Enter the title of the column. The title is displayed in the header row of the table.

5 Choose the object to be displayed in the column from the Type drop-down list.

6 To hide the column from view uncheck the Visible checkbox.

7 Enter the width of the column in pixels.

8 To make the column a fixed size, uncheck the Resizeable checkbox. If you leave this checkbox checked the column can be resized by users.

9 Configure the column using the Data, Details, Repeat, Colors and Font tabs on the Column Properties window.

Modifying a table

To reorganize columns:

1 Open the Properties window for the table and display the Columns tab.

2 Click on the column you want to move.

3 Click on the Move Up button to move the column up the list, which will display it further to the left or click on the Move Down button to move the column down the list which will display it further to the right in the table.

To remove a column:

1 Open the Properties window for the table and display the Columns tab.

2 Click on the column you want to remove.

3 Click on the Remove button.

The column is removed from the table.

Display/shape properties

7

This chapter describes the display/shape properties.

You access the properties of the current display/shape through the Properties Window. (If the Window is not visible, click .)

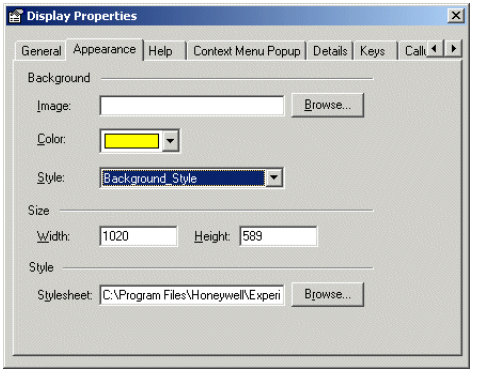
Notes

- If any object is selected, choose **Edit** > **Deselect All** so that the Properties Window shows the display/shape properties.

For details about this tab:	Go to:
Appearance	page 244
Callup Task	page 245
Custom Properties	page 246
Details	page 250
General	page 253
Help	page 258
Keys	page 261
Periodic Task	page 262
Shortcut Menu	page 263

Appearance tab

The Appearance tab contains properties that define the size and background details of the display.



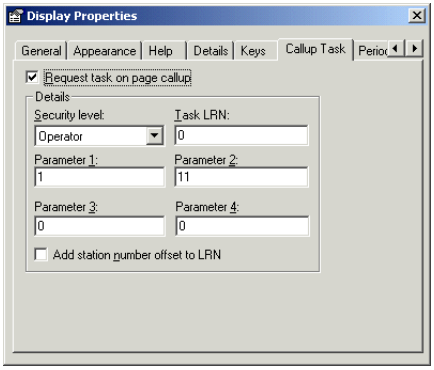
Property	Description
Background	Specifies the display's background, which can include: • An image • A color • A style from the style sheet specified in **Stylesheet**
Size	The size of the display, expressed in pixels.
Stylesheet	The style sheet used by the display. To learn about style sheets, see "About style sheets" on page 108.

Callup Task tab

The Callup Task tab specifies the *task* that is performed whenever the display is called up. (A task is any of the standard server programs or a request to an application program.)

Notes

- If you want to perform the task at regular intervals while the display is visible, use the Periodic Task tab.
- For more information about the Server Display Program (LRN 21) actions and parameters, see the *Configuration Guide*.
- For more information about displaying the LRN assigned to an application program, see the *Application Development Guide*.



Property	Description
Request task on page callup	Select this to specify a callup task.
Details	The task details, including its LRN (task identifier), parameters and the security level required to perform the task. If you select **Add station number offset to LRN**, a separate task is performed on each Station. (The number of the task performed on a particular Station is equal to the sum of the specified LRN and the Station's number.)

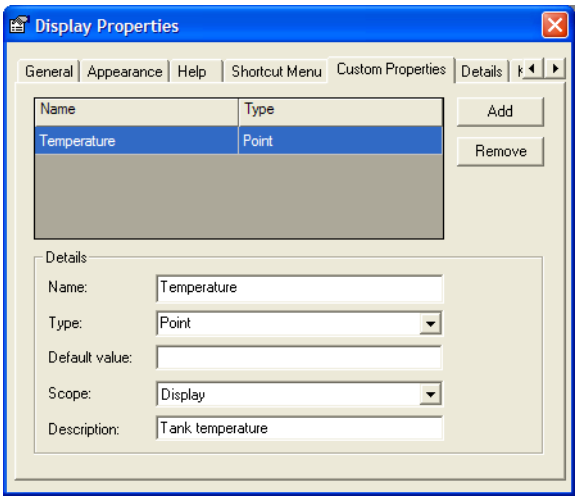
Custom Properties tab

The Custom Properties tab contains properties that are specific to each type of custom property.

For details about this type of page:	Go to:
System	page 247
Dynamic shape/shape sequence	page 249

System Custom Properties tab

The Custom Properties tab for a display allows you to define system custom properties. To learn about custom properties, see “About custom properties” on page 57.



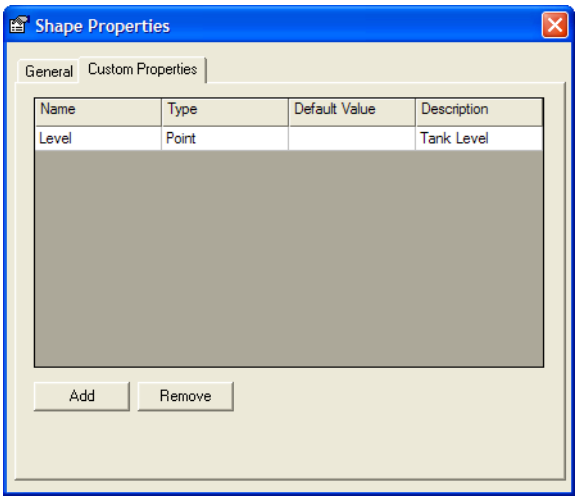
To add a custom property, click **Add**.

Column	Description
Name	The custom property’s name. The name of the custom property must be unique for the display.
Type	The custom property’s type: • `Point`. Select this if the custom property represents a point. • `Parameter`. Select this if the custom property represents a parameter. • `Value`. Select this if you want to use the custom property in scripts to store a user-defined value.
Default value	The value that is initially assigned to the custom property when the display is loaded in Station. Leave blank if you do not want to assign an initial value.

Column	Description
Scope	Specifies from where the custom property can be accessed and the life of the custom property: • `Display`. Restricts access to scripts that are only available within the current display or from the URL when calling up the display. These custom properties are stored in the display data repository and are only available when the display is loaded. • `Station`. Allows access through the automation model (for example, SSO) or from a script in the display. These custom properties are stored in the Station data repository and are available for the life of the Station.
Description	A description of the custom property.

Shape Custom Properties tab

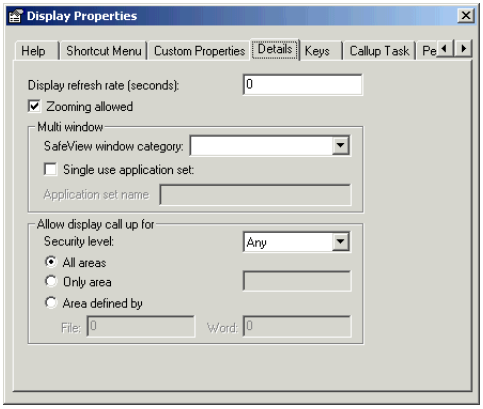
The Custom Properties tab for a shape allows you to define your own properties for the dynamic shape/shape sequence. To learn about custom properties, see “About custom properties” on page 57.



To add a custom property, click **Add**.

Column	Description
Name	The custom property’s name.
Type	The custom property’s type: • `Point`. Select this if the custom property represents a point. • `Parameter`. Select this if the custom property represents a parameter. • `Value`. Select this if you want to use the custom property in scripts to store a user-defined value. See “Using a shape custom property to store a user-defined value” on page 317.
Default value	The value that is initially assigned to the custom property when the shape is loaded in Station. Leave blank if you do not want to assign an initial value.
Description	A description of the custom property.

Details tab



Property	Description
Display refresh rate	By default, Station updates all displays at a standard rate, typically once every five seconds. This option allows you to specify a slower rate for the display. (If the display is particularly complex, slowing the refresh rate will reduce the load on your system.)
Zooming allowed	If this check box is cleared, operators cannot zoom in/out on the display.

Property	Description
SafeView window category	Only applicable if the display is to be used in conjunction with SafeView. The window category, which determines how SafeView handles the display. The standard categories are: • `HW_System_Alarm_Summary`. The Alarm Summary. • `HW_System_Detail`. A point detail display. • `HW_System_Faceplate`. A faceplate. • `HW_System_Group`. A group display. • `HW_System_Trend`. A trend. • `HW_System_Display`. A system display other than the above categories. • `NOPRIOR`. The display is managed by SafeView without appearing in the history or being re-called when a user clicks . If you need to define other categories, see “Defining SafeView window categories” on page 252.
Single use application set	Only applicable if the display is to be used by a multi-window Station. An *application set* consists of two or more related displays that must never be simultaneously displayed. The name of each application set must be unique.
Allow display call up for	**Security level** specifies the minimum security level required by a user to call up the display. If you specify an area, only Stations with the asset assigned can call up the display. **Area defined by** defines an assignable asset by a file/word combination. (In general, this option is only used by Honeywell staff.)

Defining SafeView window categories

SafeView window categories are used by SafeView to manage displays. To learn about window categories and how they are used in SafeView, see the *SafeView User's Guide*.

The window categories that appear in the **SafeView window category** list on the display's Details tab are defined in `WindowCategories.xml`. (This is an XML file, which is stored in `Program Files\Honeywell\Experion PKS\Client\HMIWeb Display Builder`.)

The structure of the XML file is as follows, where *`CategoryName1`* to *`CategoryNameN`* are the window categories that appear in the list:

```
<?xml version="1.0"?>
<categories>
  <category name="CategoryName1"/>
  .
  .
  <category name="CategoryNameN"/>
</categories>
```

The following procedure describes how to add a window category called "MyCategory" to the list of available categories.

To add a new window category:

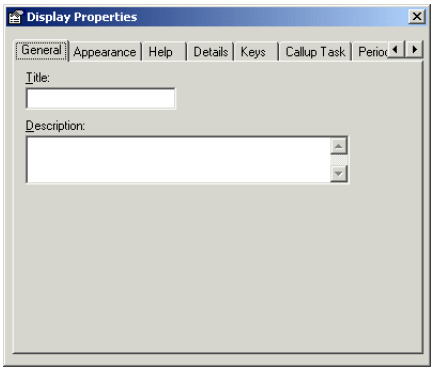
1. Using a text editor, such as Notepad, open `WindowCategories.xml`.
2. Copy an existing category entry and paste it in the required position. (The order in the XML file controls the order in which they appear in the list.)
3. Edit the pasted category entry as follows:
   ```
   <category name="MyCategory"/>
   ```
4. Save and close the file.

General tab

The General tab contains properties that are specific to each type of page.

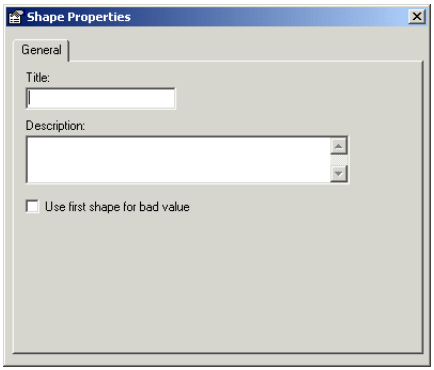
For details about this type of page:	Go to:
Display	page 254
Shape sequence	page 255
Dynamic shape	page 256
Popup	page 257

Display



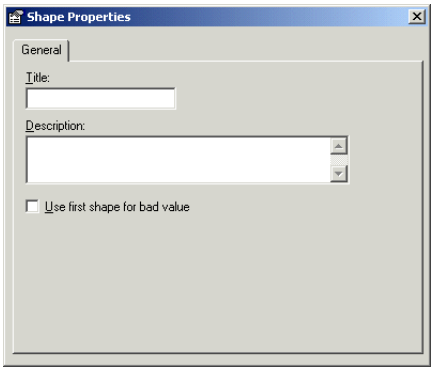
Property	Description
Title	The display's title, which appears in Station's Title Bar when a user calls up the display.
Description	Any notes or comments about the display.

Shape sequence



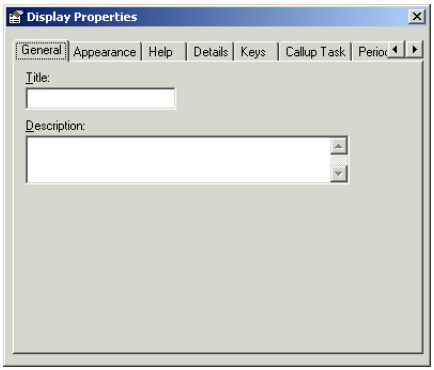
Property	Description
Title	The title of the shape file.
Description	Any notes or comments about the shape.
Stylesheet	The style sheet used by the display. To learn about style sheets, see “About style sheets” on page 108.
Use first shape for bad value	Only applicable to a shape sequence. If selected, Station displays the first shape if the server cannot determine the value of the associated point (for example, because server cannot access the point).

Dynamic shape



Property	Description
Title	The title of the shape file.
Description	Any notes or comments about the shape.
Stylesheet	The style sheet used by the display. To learn about style sheets, see “About style sheets” on page 108.
Use first shape for bad value	Not applicable to a dynamic shape.

Popup



Property	Description
Title	The title of the popup.
Description	Any notes or comments about the popup.

Help tab

The Help tab specifies the help associated with the display. Users can call up the help by pressing the help key or selecting a menu item. (Each Station must be configured so that users can assess the help. For details, see the *Configuration Guide*.)

You can specify either:

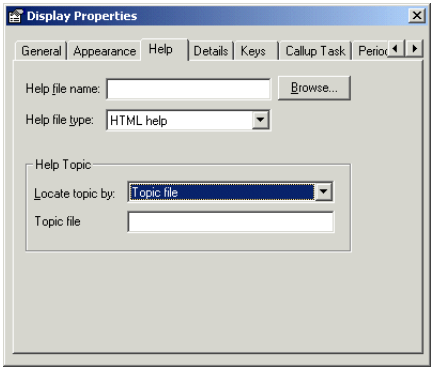
- A topic within a HTML Help file (HTML files have a "`chm`" extension)—see "Specifying a topic in a HTML Help file" on page 259
- A document such as a HTML file or Microsoft Word document—see "Specifying help other than HTML Help" on page 260

Specifying a topic in a HTML Help file

You specify a topic within a HTML Help file using either its topic number or its internal filename.

Notes

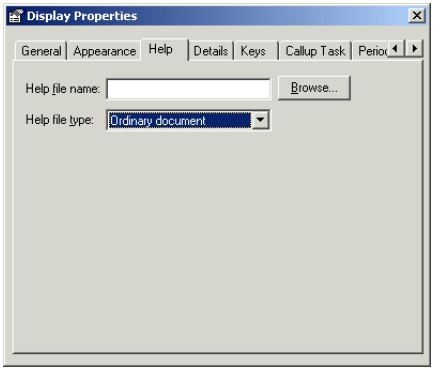
- You can create HTML Help using a variety of help authoring tools, including Microsoft HTML Help Workshop, which you can download from Microsoft's Web site.



Property	Description
Help file name	The HTML Help filename, including its `chm` extension. If the file is not in the same folder as the display, you must include the full pathname.
Help file type	Automatically set to `HTML Help` when you specify the `chm` extension.
Help Topic	The help topic, which can be specified using either its: • Internal filename. Set **Locate topic by** to `Topic file` and specify the filename of the topic in **Topic file**. • Number. Set **Locate topic by** to `Topic number` and specify the number in **Topic number**.

Specifying help other than HTML Help

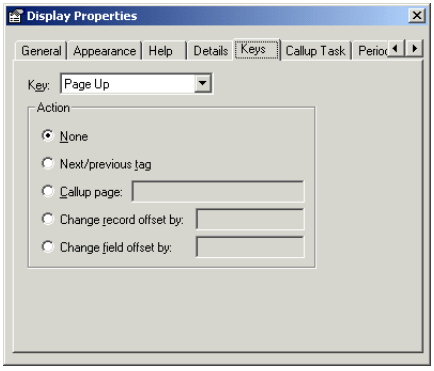
You specify help, other than HTML Help, by typing its filename. (Typical help documents include HTML pages and Word documents.)



Property	Description
Help file name	The filename of the document, including its extension. If the file is not in one of Station's display folders, you must include the full pathname.
Help file type	Automatically set to `Ordinary document` if its extension is not `chm`.

Keys tab

You can assign a range of common commands to the PAGE UP and PAGE DOWN keys, as well as the *softkeys*. (A softkey is a function key whose action is defined for each page rather than being fixed for the Station. For details about configuring softkeys, see the *Configuration Guide*.)



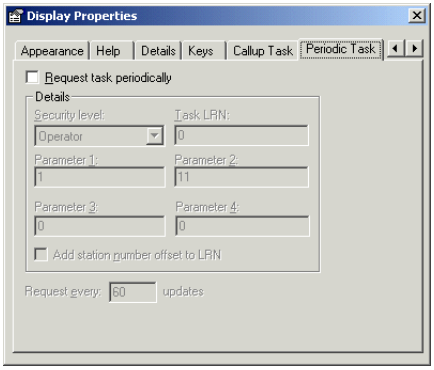
Property	Description
Key Action	A key and the associated action it performs when pressed. To assign an action to a key, select the key, and then select the action. Repeat for each key as required.

Periodic Task tab

The Periodic Task tab specifies the *task* that is performed at regular intervals while the display is visible. (A task is any of the standard server programs or an application program.)

Notes

- If you only want to perform the task when the display is called up, use the Callup Task tab.
- For more information about the Server Display Program (LRN 21) actions and parameters, see the *Configuration Guide*.
- For more information about displaying the LRN assigned to an application program, see the *Application Development Guide*.

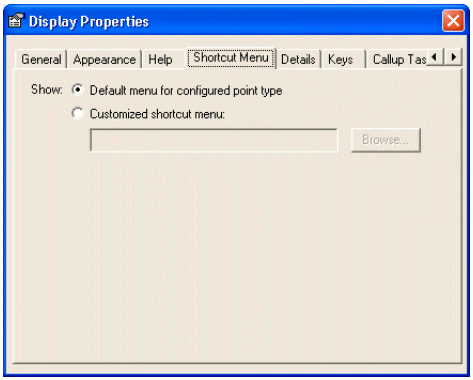


Property	Description
Request task periodically	Select this to specify a periodic task.
Details	The task details, including its LRN (task identifier), parameters and the security level required to perform the task. If you select **Add station number offset to LRN**, a separate task is performed on each Station. (The number of the task performed on a particular Station is equal to the sum of the specified LRN and the Station's number.)
Request every	The request interval, specified in display update intervals. (By default, Station updates all displays at the same rate, typically every five seconds. However, you can specify a longer interval in **Refresh rate** on the General tab.)

Shortcut Menu tab

The Shortcut Menu tab specifies the display's shortcut menu, which appears when a user right-clicks on the page.

To learn about shortcut menus, see "About shortcut menus" on page 125.



Property	Description
Show	Specifies which shortcut menu is used: • **Default menu**. Uses the default menu for a display, `sysdefault_sm.xml`. • **Customized shortcut menu**. Uses the specified custom menu.

Setting up HMIWeb Display Builder

8

This chapter describes how to set up various HMIWeb Display Builder options. It also describes keyboard and menu shortcuts.

To:	Go to:
Set up HMIWeb Display Builder's options	page 266
Learn about the diagnostic and productivity tools	page 270
Customize the toolbars	page 276
Learn about the keyboard shortcuts	page 277
Learn about the shortcut menu, which appears when you right-click	page 278

Setting HMIWeb Display Builder's options

To configure HMIWeb Display Builder's options, choose **Tools** > **Options**, and then specify the options on each tab:

- General tab
- Grids and Rulers tab
- Shapes tab (lists the folders that contain shapes)

General tab

Option	Description
Default scripting language	Specifies the default scripting language. Either `VBScript` or `JScript`.
Number of Undo/Redo Levels	Specifies the maximum number of undos/redos that you can perform. See "Undoing your work" on page 106. **Note:** HMIWeb Display Builder's memory requirements may exceed your computer's capabilities if you specify too large a number.
Recently used file list	Specifies the number of recently used files listed under the **File** menu.
Save AutoRecover info every	If you select this option, the "autorecover" version of any open display is saved at the specified frequency. If HMIWeb Display Builder crashes, it asks you whether you want to open the autorecover version when you restart it.
Change mouse cursor whenever a selectable object	If you select this option, the mouse pointer changes to when you move it over a selectable/editable object. (It does not change if, for example, you move it over a locked object.) You should consider deselecting this option if you are working on a complex display. (The extra processing involved in determining whether an object is selectable can affect performance.)
Show Intellisense auto-completion list in script editor	If you select this option, you activate the Script Editor's Intellisense feature. (IntelliSense displays information in lists and popups as you write your code. For example, it displays the available properties and methods of objects.)
Archive file format	The file format used when saving a display in archive format. Use the default file format (`Text`) unless you have a specific need to use `Binary`.
Save unarchived copy of file with archive files	If you select this option, two versions of a display are saved each time you save it in archive format: • Archive (`*.hda` or `*.hsa`) • Standard (`*.htm` or `*.sha`) Note that if you save a display in standard format, an archive version is not saved.

Grids and Rulers tab

Option	Description
Horizontal spacing Vertical spacing	The horizontal and vertical spacing, in pixels, of gridlines, expressed in pixels. The grid makes it easier to arrange display objects in a neat and consistent manner.
Color	The color of the gridlines.
Snap objects to grid	If you select this option, objects automatically align to the grid when you create, move or resize them. You can also use the button on the View toolbar to turn this option on/off.
Show grid	If you select this option, the grid is visible. You can also use the button on the View toolbar to show/hide the grid.
Show rulers	If you select this option, the rulers are visible. The rulers make it easier to arrange objects in a neat and consistent manner. You can also use the button on the View toolbar to show/hide the rulers.

Shapes tab

Option	Description
Search for shape files in	This list specifies the folders HMIWeb Display Builder searches when looking for linked shapes. See "Registering a shape folder" on page 269. If a display includes a link to a shape that is not in a listed folder, the next time you open the display, HMIWeb Display Builder will replace the shape with a rectangle containing an error message ("Unable to load shape file …"). Similarly, if a display includes an embedded shape that is not in a listed folder, you cannot upgrade it. (See "Upgrading an embedded shape" on page 99.)
By default, insert into display	Specifies the default manner in which shapes are inserted into displays: • **Contents of shape file** embeds a copy of the shape file in the display. (The recommended option because it optimizes display performance.) • **Link to shape file** inserts a link to the shape file. Note that when you insert a shape, you can override the default by selecting the other option in the **Insert Shape** dialog box.

Registering a shape folder

To add a shape folder to the list:

1 Click **Add** to add a blank line to the list.

2 Either type the folder's full path, or click the ... button and browse for the folder.

3 If necessary, use the ↑ and ↓ buttons to rearrange the order of folders in the list. (HMIWeb Display Builder searches through the folders in the order listed.)

Using the diagnostic and productivity tools

This section describes how to use the diagnostic and productivity tools in HMIWeb Display Builder.

To:	Go to:
Analyze the performance of a display	page 138
Use the HMIWeb Display Builder Assistant	page 271
Generate a Display Report	page 272
Rename points in the current display	page 274
Replacing a shape in a display.	page 275

Using the HMIWeb Display Builder Assistant

You can use the HMIWeb Display Builder Assistant to perform a range of tasks on a set of displays and shapes. Using the assistance, you can do the following tasks.

Task	Description
Rename Points	Searches for, and changes, the name of a point in each display and shape in the specified folder/subfolders.
Embed Shapes	Embeds the linked shapes in each display and shape in the specified folder/subfolders.
Upgrade Embedded Shapes	Upgrades any embedded shapes in each display and shape in the specified folder/subfolders.
Replace Shapes	Replaces shapes across multiple displays and shape files.
Archive Displays & Shapes	Creates an archive file for each display and shape in the specified folder/subfolders. To learn about archive files, see “About archive format” on page 65.
Unarchive Displays & Shapes	Creates a display/shape for each archive file in the specified folder/subfolders.

To perform a task:

1. Choose **Tools** > **Display Builder Assistant**.
2. Select the task you want to perform, enter any relevant details and then click **Run**.

Generating a Display Report

You use the Display Report tool to create a report about the objects on a display. A display report can include the following information:

- Object properties
- Point information
- Stylesheet properties
- Scripts
- Shape list
- Custom properties

To generate a Display Report:

1. Open the display that you want to generate a report about.
2. If you want to generate a report about specific objects in this display, select those objects.
3. Choose **Tools** > **Display Report**.

   The Display Report wizard is displayed.
4. If you want to generate a report about selected objects, click **Selected display objects** in the **Report on** group.
5. In the **Include** list, select the items you want included in the report. See the table below for a description of each item.

Item	Description
Object properties	This includes name, style class, behaviors, bound data, display format, script data, animation details, and hierarchy.
Point information	This includes any data bound to display objects.
Stylesheet properties	This includes the style definitions.
Scripts	This includes the name and script content of page scripts, and any scripts on display objects.
Shape list	This includes the filename, number of instances and custom properties.
Custom properties	The includes the shape custom properties.

6. Click **Next**.

   The report is displayed.

7 If you want to save the report, click **Save As**. You can save the report as an HTML page or an XML file.

8 Click **Close** when finished with the report.

Renaming a point in the current display

If the current display includes any references to a point whose name has changed, you can use the following procedure to update those references.

If the point is referenced in numerous displays, use the HMIWeb Display Builder Assistant. See “Using the HMIWeb Display Builder Assistant” on page 271.

To rename a point:

1. Choose **Tools** > **Rename Points**.
2. Type the point’s current name and new name, and then click **Rename**.

Replacing a shape in a display

You can replace a shape within a shape with another shape. The data bindings, scripting, and custom properties of the existing shape are retained when replaced with another shape.

If you want to replace a shape across multiple displays, use the HMIWeb Display Builder Assistant. See “Using the HMIWeb Display Builder Assistant” on page 271.

To replace a shape in a display:

1. Choose **Tools** > **Replace Shapes**.
2. In the **Current shape file** list, select the existing shape file name to replace.
3. In the **New shape** file box, type the path and file name to the new shape file, or click **Browse** and browse to the new shape file, click it, and then click **Open**.
4. If you want the new shape file to retain its current dimensions, select the **Resize display objects to match size of new shape** option.

   If you clear this option, the new shape will be resized to the same dimensions as the existing shape.
5. Click **Replace**.

Creating a toolbar

You can create your own toolbars that only contain the tools/menus you want.

To create a toolbar:

1. Choose **Tools** > **Customize**.
2. Click **New** on the Toolbars tab.
3. Type the toolbar's name and click **OK**.

   An empty—and small—toolbar appears. Its name also appears in the **Toolbars** list.
4. Click the Command tab.
5. Select a category that contains items (icons or menus) you want to include in your toolbar.
6. Drag each item to your toolbar. (This copies the item to your toolbar—it doesn't delete it from the original toolbar.)
7. Repeat steps 5 and 6 for items in other categories.

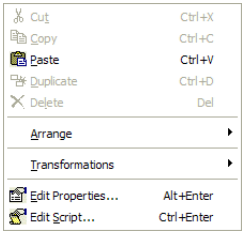
As with the other toolbars, you can show/hide your toolbar and drag it to a convenient place.

Keyboard shortcuts

Command	Shortcut
Align the left edges of selected objects	CTRL+Q
Align (horizontally) the centers of selected objects	CTRL+W
Align the right edges of selected objects	CTRL+E
Align the top edges of selected objects	CTRL+1
Align (vertically) the middle of selected objects	CTRL+2
Align the bottom edges of selected objects	CTRL+3
Bring selected object to top of stack	CTRL+PAGE UP
Bring selected object one layer up (towards the top of) the stack	SHIFT+PAGE UP
Copy selected object to the clipboard	CTRL+C
Create a new (standard) display	CTRL+N
Cut selected object to the clipboard	CTRL+X
Delete selected object	DELETE
Duplicate selected object	CTRL+D
Group selected objects	CTRL+G
Open a display	CTRL+O
Paste contents of clipboard into display	CTRL+V
Print current display	CTRL+P
Redo command that has been undone	CTRL+Y
Save current display	CTRL+S
Select all objects	CTRL+A
Send selected object to bottom of stack	CTRL+PAGE DOWN
Send selected object one layer down (towards the bottom of) the stack	SHIFT+PAGE DOWN
Show/hide Properties Window	ALT+ENTER
Show/hide Script Editor	CTRL+ENTER
Undo action	CTRL+Z
Ungroup selected group	CTRL+U

Shortcut menu

If you *right-click* (press the right mouse button), a shortcut menu appears next to the mouse pointer.



Menu item	Description
Cut	Cuts the selected object(s) to the clipboard.
Copy	Copies the selected object(s) to the clipboard.
Paste	Pastes the contents of the clipboard near the center of the display. If **Paste** is gray, it means the clipboard is empty.
Duplicate	Duplicates the selected object(s).
Arrange	Gives access to the same alignment and grouping functions as the Arrange toolbar.
Transform	Gives access to the same transform functions as the Transform toolbar.
Edit Properties	Shows the Properties Window.
Edit Script	Shows the Script Editor window.

Distributing displays

9

This chapter describes how to distribute and store displays and shape files in a manner that is appropriate to your system's setup.

To learn about:	Go to:
Storage techniques	page 280
Distribution issues (replication of displays on all the servers/Stations that need them)	page 281
Archive format (a convenient format if sending displays to remote sites)	page 65

Display storage techniques

You can use either of the following storage techniques for your displays:

- *Centralized.* All displays are stored on the server, and accessed by Stations as required
- *Distributed*. Each Station has its own copies in a local folder. (This technique reduces communication traffic and callup delays because Stations can show the displays without having to first download them.)

Distribution (replication) issues

Unless you create displays on the server and use the centralized storage technique, you need to replicate them on appropriate computer(s) and check that they work correctly.

For more information about the File Replication Service, a network copying tool, see the *Configuration Guide*.

Issue	Notes
Shapes and popups	You need to copy linked shapes and popups to folders that are registered in Station. (When asked to call up a display, Station only searches the registered folders.) For details about setting up Station, see the *Configuration Guide*. **Note:** You must manually configure the File Replication Service to ensure that it also copies supporting folders that contain linked shapes and popups.
Associated folders	Each display (and shape/popup) has an associated folder that contains various support files and graphics. These folders have the same name as the display, but with a "`_files`" extension. Copying a display will also cause the associated folder to be copied.
Point IDs and database links	Check that all dynamic objects in a display show the correct data—when configuring an object, it is easy to mistype a point ID or database address.
DSP displays	If you are also distributing DSP displays, see the *Display Building Guide*—the procedures for distributing DSP displays are slightly different.
Style sheets	Any style sheets located in paths relative to displays must be copied to the same relative path on the destination computer.

Display scripting reference

10

Display scripts allow you to extend the functionality of your HMIWeb displays. (A script is a mini-program that performs a specific task.)

For:	Go to:
Scripting basics	page 285
An introduction to the Script Editor, the tool you use to write scripts	page 327
A detailed description of each object, method and event in the HMIWeb Object Model	page 333
Example scenarios which include scripts that perform practical tasks	page 645

Notes

- If you want to write scripts for DSP displays, see the *Display Building Guide*.
- If you want to write scripts for the server, points or reports, you must write *server script*s—see the *Server Scripting Reference*.

Scripting basics

11

Attention

If you have previously written scripts for DSP displays, note that there will be substantial differences between a HMIWeb script and a functionally-identical DSP script. See “Scripting differences between HMIWeb and DSP displays” on page 324.

Scripting languages

You can write scripts in either VBScript or JScript. To specify the default scripting language, see “Setting HMIWeb Display Builder’s options” on page 266.

Neither scripting language requires extensive programming skills.

The examples given in this guide are in VBScript.

Script components

A script includes the following basic components:

- Grammar, such as `If … Then … End If`, which is provided by the scripting language
- Nouns (objects) and verbs (methods and events), which are provided by the object model

Figure 25 Typical script (grammar is shown in blue)

```
If txtMotorState.value = "0" then
    If MsgBox("Are you sure that you want to start this pump",vbYesNo) = vbYes Then
      txtMotorState.value = 1
      cmdStart.fillColor = vbGreen
      cmdStop.fillColor = vbRed
    End If
  End If
```

Help for scripting languages

This help only describes the object model (nouns and verbs). So, if you want help for the scripting language, choose either:

- **Help > VBScript Reference**
- **Help > JScript Reference**

Setting the default scripting language

You should set the default scripting language before you start writing any scripts.

To set the default scripting language:

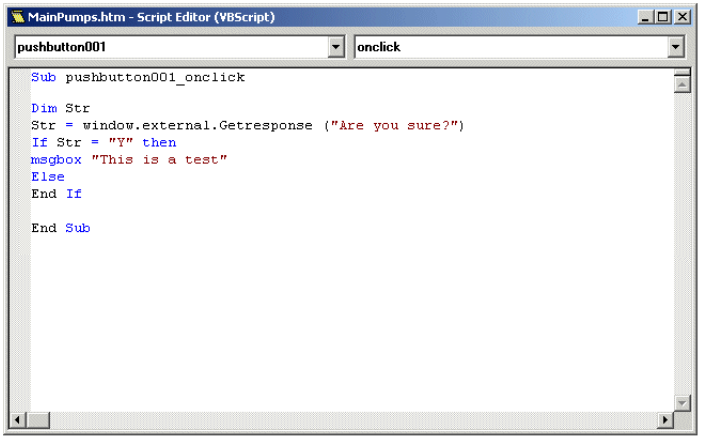
1. Choose **Tools > Options**.
2. On the **General** tab, select the required language from **Default Scripting Language**.

Events

Scripts are *event-driven*, which means that a script only runs when the associated event occurs.

The following figure shows a typical script that runs when a user clicks the object called “pushbutton001”.

Each type of event has an “on” prefix, as in `onclick` and `onafterupdate`. In addition, each script for a particular object is prefixed with the object’s name, as in `pushbutton001_onclick`.



Event bubbling

Displays support *event bubbling*, which means that an event passes up the object hierarchy until it reaches an object that has a script for that event. For example, if a user clicks an object that is part of a group, the event can bubble up from the object, to the group and finally to the page.

Event bubbling allows you to write page-level scripts that handle a particular event for all objects—this technique makes it easier to maintain and debug scripts. For example, rather than writing a separate onclick script for each object, you write one onclick script for the page. Such a script would check which object fired the event and then perform the appropriate task.

Notes

- The Object Explorer shows the display's object hierarchy.
- You can use the event object—which represents the current event—to prevent event bubbling by setting its cancelBubble property to `true`. For example, the following statement prevents event bubbling past an object called "group2":
  ```
  window.event.cancelBubble = true
  ```
- Not all events bubble through the object hierarchy. The event's description indicates if it does not bubble.

Example scenario

Using event bubbling to reduce script effort

Event firing order

HMIWeb events do not fire in set orders. The order depends on the which elements and actions are used on a page. Certain actions will cause specific events to fire, so the firing order of events is not always consistent. The following table lists some common events and the actions that trigger them.

Event	Trigger	See
OnActivate	Occurs when an element is selected.	page 564
OnBlur	Occurs when an object loses focus.	page 572
OnClick	Occurs when the mouse is clicked.	page 574
OndbClick	Occurs when the mouse is double-clicked.	page 581
OnDeactivate	Occurs when an element is deactivated.	page 582
OnFocus	Occurs when the user returns focus to Station. This event does not fire if Station already has the focus when the page is loaded.	page 588
OnKeydown	Occurs when a keyboard key is depressed.	page 589
OnKeypress	Occurs when an alphanumeric key is pressed on the keyboard.	page 590
OnKeyup	Occurs when a keyboard key is released.	page 591
OnMousedown	Occurs when the mouse button is depressed.	page 594
OnMouseEnter	Occurs when the mouse enters the page.	page 595
OnMouseLeave	Occurs when the mouse leaves the page.	page 596
OnMouseMove	Occurs when the mouse moves on the page.	page 597
OnMouseOut	Occurs when the mouse leaves the page.	page 598
OnMouseOver	Occurs when the mouse moves over an object.	page 599

Event	Trigger	See
OnMouseUp	Occurs when the mouse button is released.	page 600
OnOperatorKey	Occurs when an operator key is pressed on the Integrated Keyboard (IKB).	page 606
OnPageComplete	Occurs only when the page has completely loaded, other events may still occur even after this event has fired.	page 611
OnUnload	Occurs when one page is changed to another one.	page 632

Normally, you would expect OnActivate to fire before OnPageComplete, but in most cases, OnPageComplete usually fires first because no elements on the page have been activated yet. However, if you have a script that activates an object before the page fully loads, then OnActivate would fire first.

Scripts that are meant to run when the page has finished loading should never be executed in OnActivate since some elements may not have finished loading when this event is fired and it may fire every time the page has focus. Instead, the scripts should be moved to fire when OnPageComplate is fired to ensure they are automatically loaded.

For more information on the different events, see “Events” on page 563.

General section

The *general section* is typically used for declaring variables, and for storing general-purpose scripts and subroutines that are called by other scripts. It is also used for storing error handlers—see “Using error handlers” on page 323.

Any scripts in the general section are executed as soon as the display loads, and variables become global.

Timers

Timers provide a convenient mechanism for running scripts at regular intervals. For example, you might want to move an object across the screen, moving it ten pixels every second. To do this, you would set a timer, and increment the object's position each time interval. “Creating a moving object” on page 654 illustrates this usage in more detail.

The way in which you create timers for HMIWeb displays is different than for DSP displays. In DSP displays, timers were created using the CreateTimer method. In HMIWeb displays, timers are created using the standard HTML setInterval method.

The following example creates a 200 ms timer by using a function called “myfunction” (which is stored in the General section).

```
Sub StartBtn_onclick()
  intTimerID = window.setInterval ("myfunction()", 200)
End Sub
```

The general section contains both the function, myfunction, and the declaration for the timer ID variable, intTimerID.

```
Dim intTimerID
Function myfunction()
  .
  .
  .
End Function
```

When you no longer need the timer, you kill it as follows:

```
window.clearInterval(intTimerID)
```

When creating a timer within a shape, you may need to pass details of the shape object that created the timer to the timer callback function, so that it can refer back to the object that created it.

The following example creates a 200ms timer and the callback function receives a reference to the object that created the timer. This is done by specifying the `me.id` parameter as part of the timer creation:

```
window.setInterval ("myfunction(" & me.id & ")", 200)
Function myfunction(element)
   ' the element parameter refers to the object specified
   ' in the timer creation
   element.fillColor = ...
End Function
```

You can use the setTimeout method to evaluate an expression after a specified time period has elapsed. The following example will hide a button after 3 seconds have elapsed:

```
function fnHide(oToHide) {
window.setTimeout("fnHide2(" + oToHide.id +")", 3000);
}
function fnHide2(sID){
var o = eval(sID);
o.style.display="none";
}
</SCRIPT>
<INPUT TYPE=button Value="Count Down"
ID="oHideButton" onclick="fnHide(this)">
```

The setTimeout method is cancelled using the clearTimeout method.

Notes

- Timers with intervals of less than 100 ms are not recommended for performance reasons.

Example scenarios

Creating an animation with a shape sequence

Referencing objects

You must explicitly identify all objects (even the current one) using either of the following techniques:

- The object's name. This example changes the fill color of "rect001" to red:
  ```
  rect001.fillColor = vbRed
  ```
- VBScript's "me" keyword (or JScript's "this" keyword) if scripting the current object. This example changes the fill color of the current object to red:
  ```
  me.fillColor = vbRed
  ```
- The DHTML event object and its srcElement property if the event relates to the current object. This example changes the fill color of "rect001" to red when it is clicked:
  ```
  Sub rect001_onclick
    window.event.srcElement.fillColor = vbRed
  End Sub
  ```

Differences between the "me" keyword and object.srcElement

For most objects, the me keyword or object.srcElement are functionally equivalent.

However, in the case of a compound object, such as a group or shape, the srcElement identifies the specific object within compound object. You would use this syntax if, for example, you wanted to identify which object within a group was clicked by a user. If you only wanted to know whether the group was clicked, you should use the `me` keyword (see "Event bubbling" on page 287).

Referencing the current display and its objects

Within the current display (or dynamic shape, popup, and so on), any script attached to the Page object or any of its objects can directly reference the Page object and its top-level objects.

This example obtains the URL of the style sheet attached to the display.

```
x = page.stylesheetSrc
```

This example changes the text of a text object when a new alarm is raised.

```
Sub Page_onalarm
  textbox001.value = "Attention: new alarm"
End Sub
```

Referencing child objects within groups and shapes

You can reference child objects within a group directly by name from any script in the display. This is because all group objects are given unique names by HMIWeb Display Builder.

Referencing child objects within a shape however is slightly more complicated and depends on where you are writing the script. If the script is in the shape file itself, you can reference other objects within the shape file directly by name.

If the script is in the containing display, you can reference any object within the shape with the Objects method. For example, if "rect1" is part of "shape1", you would reference it as follows:

```
shape1.Objects("rect1")
```

Referencing child objects with the children and all collections

You can use the index number of the standard DHTML all and children collections to reference an object within a shape or group. This technique is especially useful when dealing with shapes, which are generally used in a sequential manner.

The following example shows how you could access the third object in the collection of immediate children for "Shape4":

```
Shape4.children(3)
```

The following example shows how you could access the third object in the collection of all children for "Shape4":

```
Shape4.all(3)
```

When working with groups and shapes, you may need to find out the number of objects in the group or shape. (You need the count if, for example, you want your script to step through the shapes.)

The following example shows how to obtain the count of immediate children:

```
Count = Shape4.children.length
```

The following example shows how to obtain the count of all children:

```
Count = Shape4.all.length
```

Referencing an object's parent

In general, reference an object's parent using the parentElement property. The following example shows how to reference the parent object of "alpha001":

```
alpha001.parentElement
```

In the case of a popup or faceplate, reference its parent (the object to which it is attached) as follows:

```
window.external.parentElement
```

Referencing the parent display or Station application from a popup/faceplate

Reference the parent display from one of its popups as follows:

```
window.external.parentDocument
```

Reference Station (Application object) from a popup as follows:

```
window.external.parentDocument.parentWindow.external
```

The following example shows how to call a subroutine, “MyRoutine”, in the parent display from a popup:

```
window.external.parentDocument.parentWindow.MyRoutine(value)
```

Working with shapes

Basic rules and restrictions

The following rules and restrictions apply to scripts for shapes:

- From the point-of-view of scripts, only the *current* shape exists—that is, the shape which is currently visible in the display. This means that a script needs to check whether a shape is current before attempting to control it.
- Events are only passed to the current shape.
- You use the Shape object to control the current shape. For example, you use the shape's value property to set the current shape:
  ```
  Shape1.Value = 2
  ```

Using general section functions

You can improve the performance of shape-based scripts by using general section functions in the shape to provide the required functionality, and calling those functions with single-line scripts from the event scripts for the relevant objects.

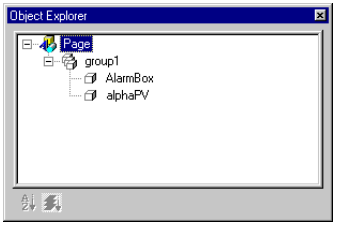
The performance improvements are particularly significant if there are multiple instances of the shape in a display. (This occurs because the functions are "loaded" into the general section of the display and are called by all instances of the shape.)

Notes

- The name of each function must be unique at both the shape and display level. (It is a good idea to develop a naming convention to ensure that there is never a conflict. For example, you could use the shape's filename as the prefix for all function names in that shape.)
- The functions cannot use the "me" keyword or the id parameter of a collection to reference an object because the function does not belong to a particular object or collection. The following example shows the coding technique used to work around this restriction.

Example

You have a shape that contains two objects (excluding the group and page). You want the color of the object called "AlarmBox" to change to red when the PV of the point associated with the alphanumeric, "alphaPV", is greater than 50.



You write the following script for the alphanumeric's onupdate event:

```
Sub alphaPV_onupdate
  call inline_alphaPV_onupdate(me)
End Sub
```

You write the following function called "inline_alphaPV_onupdate" in the general section. Note the use of:

- `parentElement.parentElement`, to move up the object hierarchy and obtain the shape object
- The Objects method to obtain the AlarmBox

```
sub inline_alphaPV_onupdate(elem)

  dim varAlarmBox
  ' Walk up the DOM to obtain the shape object & use the Objects
   method
  ' on the shape object to obtain the AlarmBox element
  set varAlarmBox =
   elem.parentElement.parentElement.Objects("AlarmBox")

  if (elem.value > 50) then
    varAlarmBox.fillColor = vbRed
  else
    varAlarmBox.fillColor = vbGreen
  end if

end sub
```

Working with popups and faceplates

Scripting in a multi-window environment

When you are working in a multi-window environment the only faceplate scripting properties that are supported are:

```
window.external.parentDocument
window.external.parentElement
```

Closing a popup/faceplate from a script on the popup

Identify the popup/faceplate with `window.external`. For example, to close the popup:

```
window.external.close
```

Referencing the object to which the popup/faceplate is attached

Reference the object to which the popup/faceplate is attached as follows:

```
window.external.parentElement
```

Referencing the display or Station application from a popup/faceplate

Reference the display from a popup as follows:

```
window.external.parentDocument
```

Reference Station (Application object) from a popup as follows:

```
window.external.parentDocument.parentwindow.external
```

The following example shows how to call a subroutine, “MyRoutine”, in the parent display from a popup:

```
window.external.parentDocument.parentWindow.MyRoutine(value)
```

Changing the size of a popup window

You can change the popup window’s size. For example, to set the window’s height to 500 pixels high and 300 pixels wide:

```
window.external.dialogHeight = 500
window.external.dialogWidth = 300
```

Note that the minimum height you can set with scripting is 100 pixels.

Changing the size of popup window's content

You can change the size of the popup window's content—in effect, zoom in on (or out from) the content. For example, to set the content's height to 500 pixels high and 300 pixels wide:

```
page.style.height = 500
page.style.width = 300
```

Accessing Station-related properties in page-level scripts

Page-level scripts can access Station-related properties using the:

- Station object, which is Station's root object and is accessed using `window.external.Parent`
- AppWindow object, which represents the Station window holding the current display and is accessed using `window.external.AppWindow`
- RuntimeStatus object, which represents Station's status details and is accessed using `window.external.Parent.RuntimeStatus`

Examples

This example shows how to obtain the ID of the currently selected point. (An Application object property)

```
x =
window.external.Parent.ActiveWindow.CurrentPage.CurrentPointID
```

This example shows how to obtain the time that appears in the Time box of the Status Line (a RuntimeStatus object property).

```
x = window.external.Parent.RuntimeStatus.Time
```

Station runtime status

The Station RuntimeStatus object allows you to call up or refer to properties of Station that are normally displayed on the status bar. This object allows you to incorporate Station runtime status into your scripts. This is a useful way of getting crucial information to the operator quickly and efficiently.

The following script can be attached to an alphanumeric object to display the current operator level to the user:

```
Sub alpha002_onclick
if window.external.Parent.RuntimeStatus.SecurityLevelAcronym <>
   "Mngr" then
msgbox "You need a Manager security level to view this
   faceplate. Your security level is " +
   window.external.Parent.RuntimeStatus.SecurityLevelAcronym +"
   "
end if
End Sub
```

The following script can be added to a pushbutton object to display to the user what type of security is configured on the currently connected Station, such as Station-based or Operator-based security:

```
Sub pushbutton003_onclick
   If window.external.Parent.RuntimeStatus.SecurityType = 0 then
      msgbox "Connected Station is running Station Based
   Security"
   else
   If window.external.Parent.RuntimeStatus.SecurityType = 1 then
      msgbox "Connected Station is running Operator Based
   Security"
   else
   If window.external.Parent.RuntimeStatus.SecurityType = 2 then
      msgbox "Connected Station is running Operator Based
   Security with Windows Users"
   else
   If window.external.Parent.RuntimeStatus.SecurityType = 3 then
      msgbox "Connected Station is running Operator Based
   Security with Windows Users and Signon Manager"
   else
   If window.external.Parent.RuntimeStatus.SecurityType = 4 then
      msgbox "Connected Station is running Operator Based
   Security with Windows Users and Single signon"
   else
   end if
   end if
   end if
   end if
   end if
End Sub
```

These scripts are useful for Stations which are consistently used by different operators, allowing them to get information about the Station and server not normally displayed to the user, such as the Station security type.

Setting and returning colors

Setting colors

You can set colors using:

- A HTML named color. For example:
  ```
  rect002.fillColor = "deeppink"
  ```
  For details about named colors, see `http://msdn.microsoft.com/library/default.asp?url=/workshop/author/dhtml/reference/colors/colors.asp`.
- A HTML-formatted 24-bit RGB value. The format is: #*rrggbb*, where *rr*, *gg* and *bb* are the hexadecimal values for the red, green and blue components. This example sets the color to blue:
  ```
  rect002.fillColor = "#0000FF"
  ```
- A VBScript color constant. For example:
  ```
  rect002.fillColor = vbBlue
  ```
  See “VBScript color constants” on page 303.

You can also set the color to “transparent”, for example:

```
rect002.fillColor = "transparent"
```

Returning colors

Colors are returned as HTML-formatted 24-bit RGB values, or as `transparent` if the color is “transparent”.

For example, if the fill color of rect002 was HTML “red”, the following code would set CurrentColor to `#FF0000`:

```
CurrentColor = rect002.fillColor
```

VBScript color constants

The VBScript color constants are:

- `vbBlack`
- `vbRed`
- `vbGreen`
- `vbYellow`
- `vbBlue`
- `vbMagenta`
- `vbCyan`
- `vbWhite`

Reading and writing to point parameters

Scripts can read and write to point parameters providing you have added them to an object's Script Data tab. (To learn how to do this, see .)

Adding a point parameter to the Script Data tab also adds a corresponding set of point-related properties and methods, such as DataValue, to the object. For a description of these properties, see "Properties and methods of points on the Script Data tab" on page 306.

Notes

- Adding a point parameter to the Script tab of one object makes the parameter accessible to all scripts in the display.
- The point-related properties are automatically updated whenever there is a change in a parameter value. See the "ondatachange event" on page 579.
- To learn how to write scripts for dynamic shapes that access point parameters, see "Accessing the point mapped to a shape custom property" on page 310.

Examples

These examples show how to read and write to the parameter shown in the following figure.

To read the parameter's value:

```
x = pushbutton001.DataValue("poista218.OP")
```

To write asynchronously to the parameter:

```
pushbutton001.DataValue("poista218.OP") = 1
```

To write synchronously to the parameter, and then check whether the write was successful:

```
check = pushbutton001.SetDataValueSynchronous("poista218.OP", 1)
If check = True Then
  .
  .
Else
  .
  .
End If
```

Example scenarios

Using a button to start a pump

Creating a jogging control

Properties and methods of points on the Script Data tab

These properties only exist for an object if one or more point parameters are listed on its Script Data tab.

To learn how to use these properties in scripts, see “Reading and writing to point parameters” on page 304.

Property	Description
`AlarmSeverity (`*`point.parameter`*`)`	Returns the severity of the alarm for the specified point, which can be: `3` = Urgent `2` = High `1` = Low `0` = No alarm
`DataAlarm(`*`point.parameter`*`)`	Returns the alarm status of the specified point parameter.
`DataChanged(`*`updatetype`*`/`*`point.parameter`*`)`	Returns `True` if: • any point parameters with the specified *`updatetype`* have been updated during the current update cycle – or – • the specified point parameter has been updated during the current update cycle otherwise returns `False`. *`Updatetype`* corresponds to the parameter’s **Update** property (specified on the object’s Script Data tab), for example: `Default` or `On Demand`. The RequestDemandData method is used to update parameters for which this property is set to `On Demand`.
`DataExists(point.parameter)`	Returns True if a reference to the specified script data parameter exists, otherwise returns False.
`DataValue(`*`point.parameter`*`)`	Returns or asynchronously sets the value of the specified point parameter. (“Asynchronously sets” means that the script continues, without waiting to check whether the value is written to the field device/controller.)
`InAlarm(`*`point.parameter`*`)`	Returns `True` if the specified point is in alarm, otherwise returns `False`.
`IsNaN(`*`point.parameter`*`)`	Returns `True` if the value of the specified point parameter is not a number (NaN), otherwise, returns `False`.
`QualityBad(`*`point.parameter`*`)`	Returns `True` if the quality of the specified point parameter is bad. Otherwise, it returns `False`.

Property	Description
QualityConfigError (*point.parameter*)	Returns True if the name of the specified point or parameter is incorrect. Otherwise, it returns False.
QualityDeviceFailure*(point.parameter)*	Returns True for a device failure bad quality, otherwise it returns False.
QualityGood(*point.parameter*)	Returns True if the quality of the specified point parameter is good. Otherwise, it returns False.
QualityLastKnown (*point.parameter*)	Returns True if the server has stopped. Otherwise, it returns False.
QualityNonSpecific *(point.paramter)*	Returns True for a non-specific bad quality, otherwise it returns False.
QualityOutOfService (*point.parameter*)	Returns True if the server is running as a backup (only applicable to a redundant server system). Otherwise, it returns False.
SetDataValueSynchronous (*point. parameter*, *value*)	Synchronously sets the specified point parameter to the specified value. ("Synchronously sets" means that the script waits until the value has been written to the field device/controller before continuing.) Returns True if the write is successful, otherwise returns False. Note that the SetDataValueSynchronous method will not return until the new value has been successfully written or an error occurs. This could result in station appearing to "lock up" while waiting for the value to be written. This method should only be used where synchronous behavior is essential.
UnacknowledgedAlarm (*Point.Parameter*)	Returns True if the specified point has unacknowledged alarms, otherwise returns False.

Debugging scripts that access points

Points listed on the Script Data tab have no visual representation on a display. Consequently, in order to check that a script which accesses points is operating correctly, you should (during development) consider:

- Writing point values to alphanumerics or message boxes
- Checking the point values at runtime using the script debugger

Using custom properties in scripts

Scripts can use custom properties in several ways:

- If a custom property is mapped to a point, scripts can:
  - Access the point. For shape custom properties, see “Accessing the point mapped to a shape custom property” on page 310. For system custom properties, see “Accessing the point mapped to a system custom property” on page 312.
  - Re-map the custom property to another point at run time. For shape custom properties, see “Re-mapping a shape custom property to another point at run time” on page 314. For system custom properties, see “Re-mapping a system custom property to another point at run time” on page 315.
- If a custom property is not mapped to a point, scripts can use it to store user-defined values—see “Using a shape custom property to store a user-defined value” on page 317. For system custom properties, see “Using a system custom property to store a user-defined value” on page 318.

To learn about custom properties, see “About custom properties” on page 57.

Accessing the point mapped to a shape custom property

You can write scripts that use a shape's custom properties to access whichever points are currently mapped to the custom properties.

Notes

- In order to use custom properties in scripts, you must first add them to an object's Script Data tab. (You can select any object, but it is generally more convenient to select the object to which you want to attach your script(s).)

Example

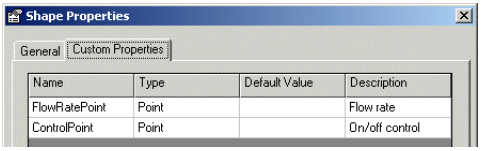
This example uses an ondatachange script to obtain the values of the point parameters currently mapped to custom properties "ControlPoint" and "FlowRate", and perform a task based on the value of "ControlPoint".

(Steps 1 to 4 summarize the main steps involved in creating a dynamic shape. For detailed instructions, see "Creating a dynamic shape" on page 46.)

To create the shape and write the script:

1 Create the shape.

2 Specify the shape's custom properties.

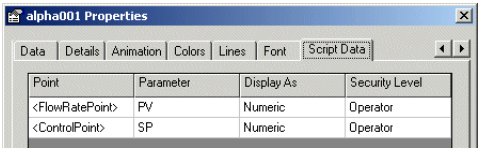


3 Add objects to the shape.

4 Assign the custom properties to the dynamic objects.

5 Select an appropriate object (in this example, "alpha001") and in the Behaviors tab of the Properties Window, select **Script Data**—this adds the Script Data tab.

6 Click the Script Data tab and add the custom properties. (The angle brackets indicate that a "point" is really a custom property.)



7 Open the Script Editor and write the following script.

```
Sub alpha001_ondatachange
  'Obtain the point parameter values
  PumpStatus = alpha001.DataValue("ControlPoint.SP")
  Litres = alpha001.DataValue("FlowRatePoint.PV")

  'Now check the pump's status
    If PumpStatus = 1 then
      'Do something
    Else
      'Do something else
    End If
End Sub
```

Accessing the point mapped to a system custom property

You can write scripts that use system custom properties to access whichever points are currently mapped to the custom properties.

Notes

- In order to use custom properties in scripts, you must first add them to an object's Script Data tab. (You can select any object, but it is generally more convenient to select the object to which you want to attach your script(s).)

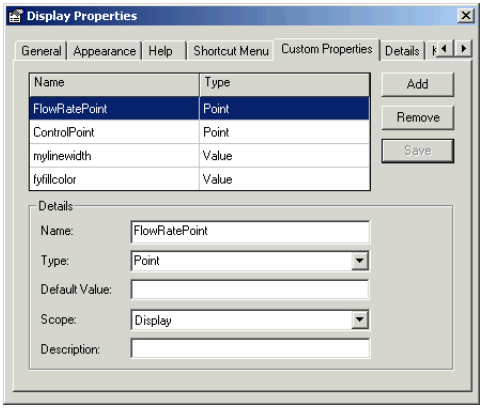
Example

This example uses an ondatachange script to obtain the values of the point parameters currently mapped to custom properties "ControlPoint" and "FlowRatePoint", and perform a task based on the value of "ControlPoint".

To insert the object and write the script:

1 Create the object.

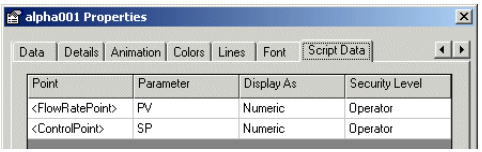
2 Specify the display's custom properties.



3 Assign the custom properties to the object.

4 Select an appropriate object (in this example, "alpha001") and in the Behaviors tab of the Properties Window, select **Script Data**—this adds the Script Data tab.

5 Click the Script Data tab and add the custom properties. (The angle brackets indicate that a "point" is really a custom property.)



6 Open the Script Editor and write the following script.

```
Sub alpha001_ondatachange
  'Obtain the point parameter values
  PumpStatus = alpha001.DataValue("ControlPoint.SP")
  Litres = alpha001.DataValue("FlowRatePoint.PV")

  'Now check the pump's status
    If PumpStatus = 1 then
      'Do something
    Else
      'Do something else
    End If
End Sub
```

Re-mapping a shape custom property to another point at run time

When you insert a dynamic shape into a display, you map each custom property to a particular point. However, you can write scripts that re-map the custom properties to other points at run time.

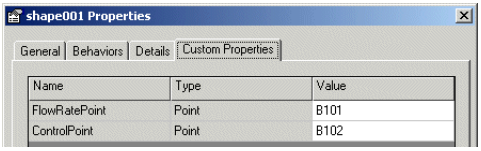
Example

This example shows how to write a script for the onclick event of a pushbutton, which re-maps a shape's custom properties to two other points. (In this example, the custom properties are initially mapped to "B101" and "B102" when the display is called up. The pushbutton's onclick script re-maps them to "B103" and B104".)

To insert the shape and write the script:

1 Insert the dynamic shape into the display.

2 Assign its custom properties to the appropriate points in the normal manner. (In effect, these are the default points that are used by the display until the pushbutton is clicked.)



3 Add the pushbutton, open the Script Editor and write the following script for the pushbutton's onclick event.

```
Sub pushbutton001_onclick()
  shape001.autoReloadContent = False
  shape001.SetCustomProperty("Point", "FlowRatePoint", "B103")
  shape001.SetCustomProperty("Point", "ControlPoint", "B104")
  shape001.ReloadContent
End Sub
```

Re-mapping a system custom property to another point at run time

When you insert an object into a display, you map each custom property to a particular point. However, you can write scripts that re-map the custom properties to other points at run time.

Example

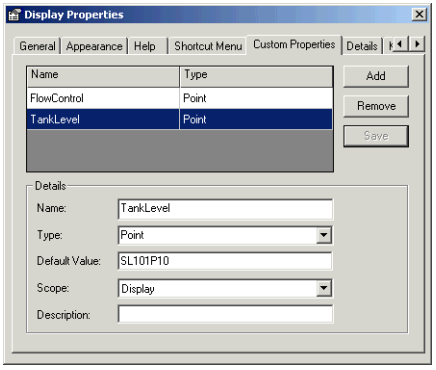
This example shows how to write a script for the onclick event of a pushbutton, which re-maps the system custom properties to two other points. The scope of the system custom properties has been set to `Station`. (In this example, the custom properties are initially mapped to “FIC100” and “SL101P10” when the display is called up. The pushbutton’s onclick script re-maps them to “FIC110” and “SL101P20”.)

To improve the performance if you are changing a lot of properties values, you should set stationDataRepository.autoReloadContent to True.

To insert the object and write the script:

1 Insert the object into the display.

2 Assign the custom properties to the appropriate points in the normal manner. (In effect, these are the default points that are used by the display until the pushbutton is clicked.)



3 Add the pushbutton, open the Script Editor and write the following script for the pushbutton's onclick event.

```
Sub pushbutton001_onclick()
  window.external.stationDataRepository.autoReloadContent =
   False
  window.external.stationDataRepository.PutValue("FlowControl",
   "FIC110")
  displayDataRepository.PutValue("TankLevel", "SL101P20")
  displayDataRepository.autoReloadContent = True
  End Sub
```

Using a shape custom property to store a user-defined value

Scripts can use a custom property to store a user-defined value.

You can use either of the following two techniques to read the value of a custom property:

- Use the GetCustomProperty method.
- Use the following syntax (although this syntax is more performant, it can only be used by scripts within the shape):

  {%*custompropertytype*::*custompropertyname*%}

Use the SetCustomProperty method to change the value of custom property.

Example

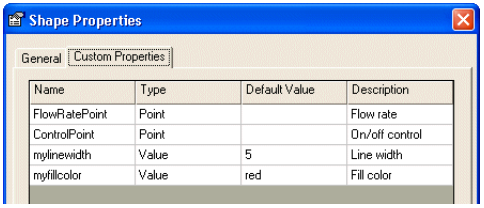
This example shows how to write an onclick script for a rectangle (“rect001”) that uses custom properties to set its line width and fill color. The example demonstrates how to use both techniques to obtain the value of a custom property.

(Steps 1 to 3 summarize the main steps involved in creating a dynamic shape. For detailed instructions, see “Creating a dynamic shape” on page 46.)

To write scripts that use the values of shape custom properties:

1 Create the shape.

2 Specify the shape’s custom properties. In the case of custom properties used to store user-defined values, set **Type** to `Value`.



3 Add objects to the shape.

4 Open the Script Editor and write the script for the rectangle’s onclick event.

```
Sub rect001_onclick
  'Obtain a custom property value from the shape element
  rect001.LineWidth =
   group001.parentElement.GetCustomProperty("Value",
   "mylinewidth")
  'Obtain a custom property value directly
  rect001.fillColor = "{%Value::myfillcolor%}"
End Sub
```

Using a system custom property to store a user-defined value

Scripts can use a custom property to store a user-defined value.

You can use the GetValue method to read the value of a custom property:

Use the PutValue method to change the value of custom property.

Considerations

- You cannot use {%*custompropertytype*::*custompropertyname*%} for system custom properties.

Example

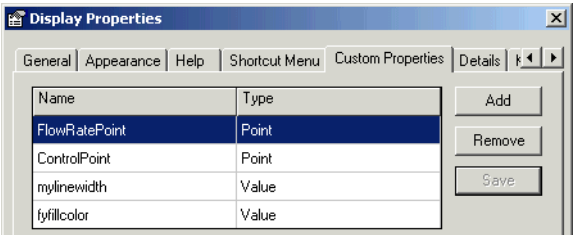
This example shows how to write an onclick script for a rectangle (“rect001”) that uses custom properties to set its line width and fill color. The example demonstrates how to use both techniques to obtain the value of a custom property.

(Steps 1 to 3 summarize the main steps involved in creating a standard display. For detailed instructions, see “Creating a standard display” on page 45.)

To write scripts that use the values of system custom properties:

1 Create the display.

2 Specify the display’s custom properties. In the case of custom properties used to store user-defined values, set **Type** to `Value`.



3 Add objects to the display.

4 Open the Script Editor and write the script for the rectangle’s onclick event.

```
Sub rect001_onclick
  'Obtain a custom property value
  rect001.LineWidth =
   window.external.stationDataRepository.GetValue("mylinewidth")
  'Obtain a custom property value directly
End Sub
```

Scripting in a multi-window environment

In a multi-window Station, it is possible for scripts on one display to refer to objects on another display. To support this functionality, Station provides a specialized collection, called StationWindows, which provides access to each Station window.

For more information, see “StationWindows object” on page 405.

Transferring data between windows

You can use custom events to share or transfer data between different windows in a multi-window environment.

The following example shows how to use the RaiseCustomEvent method to transfer data between windows. In this example two files have been loaded - “send.htm” and “receive.htm”.

The following script in the display “send.htm” fires the custom event which is received by the display file “receive.htm”:

```
Sub pushbutton001_onclick
window.external.RaiseCustomEvent “ExampleCustomEventNameSpace”,
   “ExampleCustomEvent”, “Test String”
End Sub
```

The following event handler in the file “receive.htm” receives and processes the custom event:

```
Sub Page_oncustomevent
Dim strNamespace
strNamespace = window.event.getAttribute(“OnCustomEvent_
   NameSpace”,0)
if (strNamespace = “ExampleCustomEventNameSpace”) then
    textbox002.value = strNamespace
    textbox003.value = window.event.getAttribute(“OnCustomEvent_
   EventObject”, 0)
    textbox001.value =
   window.event.getAttribute(“OnCustomerEvent_URN”,0)
end if
End Sub
```

URN attribute of the event refers to the name of the custom event and namespace attribute refers to the name of the group of events that this event belongs to. The if statement in the event handler checks whether the desired event has been received, and if so, it displays the event details.

Writing Station-level scripts

If you want to write a Station-level script—that is, a script associated with the Application object—you have two options. You can either create a *Station Scripting Object* or you can use the Script Editor in Station.

Using Station Scripting Objects

A Station Scripting Object (SSO) is an ActiveX control that attaches Station-level scripts to a Station. For example, you would create an SSO if you wanted to write a script for the OnConnect event.

A sample Visual Basic project for an SSO, `SSO.vbp`, provides the framework for implementing an SSO. (The project and associated components are zipped into `SSO_Sample.zip`, which is in `Program Files\Honeywell\Experion PKS\Client\Station\Samples`.)

For details about creating an SSO, see the *Application Development Guide*.

Using Script Editor in Station

You write scripts for Station (that is, scripts associated with the Application object) in essentially the same way as you write them for a display. However, because Station-based scripts are stored in a separate file, you edit them from within Station.

For details about writing scripts for Station, see “Using the Script Editor in Station” on page 328.

Specifying dimensions and position

HTML allows you to specify the dimensions and position of an object in several ways, including pixels, inches and centimeters. However, we recommend that you always use pixels. (This is the unit used by HMIWeb Display Builder.)

For example:

```
alpha1.style.pixelLeft=100
alpha1.style.pixelTop=200
alpha1.style.pixelWidth=250
alpha1.style.pixelHeight=80
```

Attention

A string is returned if you use `.style.left`, `.style.top` and so on to obtain a value. For example: “100px” or “3cm”.

Comparing strings

If you do not know the capitalization of a server-derived string, you should use VBScript's StrComp function. (For example, a point's status might be "Open", "OPEN" or "open".)

For example, instead of this:

```
If alpha1.value = "open" Then
```

Use this:

```
If StrComp(alpha1.value, "open", vbTextCompare) = 0 then
```

Using error handlers

You can use error handlers to help debug errors in your scripts.

If an error handler is successfully called and has no return value, or if the return value is true, then Station assumes the error has been handled.

If an error handler is not successfully called (because, for example, it does not exist), or if the return value is false, Station assumes the error was not handled. In this case a message appears, detailing the error properties, and further error notifications are suppressed in the normal manner.

Notes

- Error handlers must be stored in the general section.
- The name of the current error handler is defined by the ScriptErrorHandler property of the Page object. For example:
  ```
  page.ScriptErrorHandler = "MyErrorHandlerFunction"
  ```
- Error handlers do not accept parameters, and return a boolean value indicating whether the error has been handled or not.
- In order for script error handlers to receive notifications of script errors, Internet Explorer must be set up with script debugging disabled. (To do this, choose **Tools > Internet Options**, click the **Advanced** tab and under **Browsing**, clear the **Disable script debugging** check box.)
- If a script has set "On Error Resume Next", any script errors in the script will not be received by the error handler on the page.

Error handler prototypes

VBScript

```
sub MyErrorHandlerFunction
  ...
end sub

function MyErrorHandlerFunction
  ...
  MyErrorHandlerFunction = true
end function
```

JScript

```
function MyErrorHandlerFunction()
{
  ...
  return true;
}
```

Scripting differences between HMIWeb and DSP displays

This topic is only applicable if you have written scripts for DSP displays—DSP displays are created with Display Builder, not HMIWeb Display Builder.

Because HMIWeb and DSP displays use different object models, there will be substantial differences between a HMIWeb script and a functionally-identical DSP script.

No inherent object

In DSP scripts, you do not need to specify the object name in a script if the script is attached to that object. The following script changes the fill color of "rect1" to red:

```
sub rect1_onclick
  FillColor = vbRed
end sub
```

In HMIWeb scripts, all objects must be identified, using either of the following techniques:

- Explicitly identify the object:
  ```
  rect1.fillColor = vbRed
  ```
- Use VBScript's "me" keyword (The JScript equivalent is the "this" keyword.):
  ```
  me.fillColor = vbRed
  ```
- Use the event object and its srcElement property:
  ```
  window.event.srcElement.fillColor = vbRed
  ```

Station-level scripts

In HMIWeb, Station-level scripts are implemented using Station Scripting objects. See "Writing Station-level scripts" on page 320.

No page-level Dictionary object

DSP scripts have a Dictionary object on the display.

HMIWeb scripts, however, only have one at the application level.

Different object hierarchy

DSP scripts can navigate the object hierarchy via the Objects collection.

HMIWeb scripts can use a range of collections, the most useful being `all` and `children`, and possibly also `tags`.

The `all` collection allows you to iterate through all the elements of a display, while the `children` collection allows you to iterate through the children of an object, such as a group or a div.

The `tags` method returns a collection of all objects of a given tag, which can be used to perform an operation on all objects of a single type.

Use of the style object

DSP scripts access an object's properties directly, for example:

```
object1.left = 100
```

HMIWeb scripts access the properties of some types of object through the style object:

```
object1.style.pixelLeft = 100
```

The style object has many properties, and governs position, layout (such as margins), visibility and font attributes for HTML elements.

Confirmation of changes/clicks

With DSP scripts it is possible to use the CancelClick and CancelDoubleClick methods to cancel an event. For example, the script can check the user's security level before passing the event to the server.

With HMIWeb scripts you can use the CancelChange method and CancelEvent method on dynamic objects (objects that are linked to the database).

Useful Web sites and reference books

Microsoft Web site

Microsoft provides comprehensive documentation for the DOM and DHTML at its Web site. At the time of writing, the address is:

```
http://msdn.microsoft.com/library/default.asp?url=/workshop/
author/dhtml/reference/dhtml_reference_entry.asp
```

Books

The following books may be of use, especially if you are new to DHTML and scripting:

- *IE5 Dynamic HTML Programmer's Reference*
  Brian Francis, Alex Homer and Chris Ullman
  Wrox Press
  ISBN 1-861001-74-6
- *HTML: The Complete Reference*
  Thomas A. Powell
  McGraw Hill
  ISBN 0-07-211977-2
- *Teach Yourself Dynamic HTML In A Week*
  Bruce Campbell and Rick Darnell
  SAMS.Net Publishing
  ISBN 1-57521-335-4

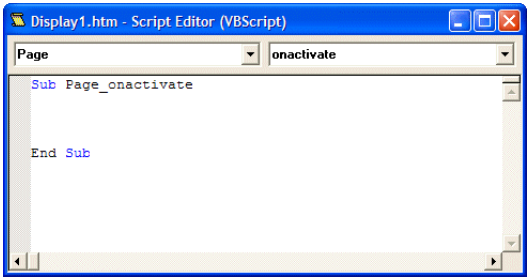
Using the Script Editor

12

You use the Script Editor to write scripts for objects and the display itself. (The scripts are stored as part of the display.)

If the Script Editor is not visible, click . (Alternatively, right-click and choose **Edit Script** from the shortcut menu.)

The Script Editor is *modeless*, which means that your script is saved to memory as soon as you select another object or event. (However, your changes are only saved to disk when you save the display.)



Using the Script Editor in Station

Writing Scripts for Station

You write scripts for Station (that is, scripts associated with the Application object) in essentially the same way as you write them for a display. However, because Station-based scripts are stored in a separate file, you edit them from within Station.

Attention

A Station can have several script files. However, only one file at a time is active: the one listed in the **Location** box of the **Station Setup** dialog box.

To write/edit scripts for Station:

1. Start Station in the normal manner.
2. Select **Station** > **Connection Properties**.
3. If the script file you want to edit is not shown click Browse. Select the script file you want and then click **Open**.
4. Click **Edit** to open the Station Script Editor.
5. Select either Application or General.
6. If you selected the Application object, select the event you want to trigger your script.
7. A skeleton event handler appears in the Script window. Click below the Sub statement and start writing your script.

Activating Intellisense

The Script Editor's Intellisense feature makes coding easier because it displays information, such as properties and methods, in lists and popups as you write your code.

To activate Intellisense:

1. Choose **Tools** > **Options**.
2. On the General tab, select **Show Intellisense auto-completion list in script editor**.

Writing scripts

To write a script:

1 Click to open the Script Editor.

2 Select an object from the left-hand (**Object**) list. This shows every object, including `Page` (the display itself) and `General` (for general-purpose scripts).

3 Select the event that will run your script from the right-hand (**Event**) list.

4 A skeleton event handler appears in the Script Editor. Click below the Sub statement and start writing your script for that event.

5 Repeat steps 2 to 4 if you want to write more scripts.

Notes

- Objects that already have scripts are shown in bold in the **Object** list. Similarly, events with scripts for the selected object are shown in bold in the **Event** list.
- The Script Editor supports the **Cut** (CTRL+X), **Copy** (CTRL+C) and **Paste** (CTRL+V) functions.
- Changes to scripts are saved to memory as soon as you select another object or event. (However, they are only saved to disk when you save the display.)
- The basic syntax of your script is checked when you select another object or event. Any syntax errors are displayed in a message box.
- Right-clicking calls up the shortcut menu, which contains several useful commands, such as **Indent**.

Finding/replacing strings in scripts

Choose **Edit > Find** to search the display's scripts for instances of a specified string.

If you want to replace one string with another, choose **Edit > Replace**.

HMIWeb Object Model reference

13

HMIWeb displays are customized DHTML (Dynamic HTML) pages that are based on the Web-standard Document Object Model (DOM).

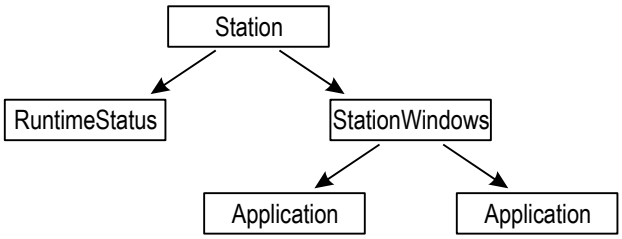
This means that, in general, display scripts are very similar to scripts used in standard Web pages.

Attention

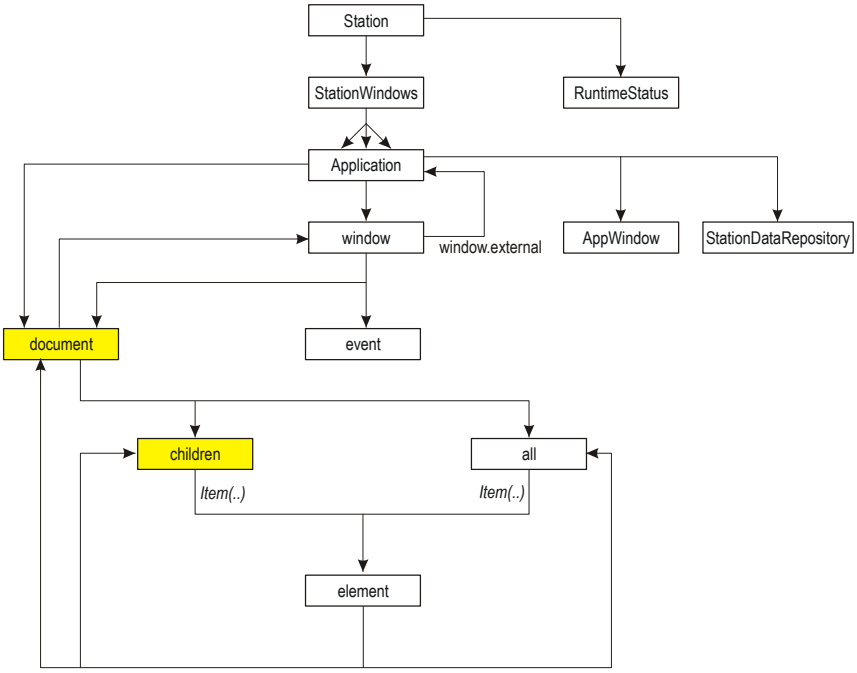
Because of the sheer size and complexity of the DOM, this help only describes objects, methods and events that are either specific to HMIWeb displays, or are particularly important.

Overview of the HMIWeb Object Model

The following figure shows the levels of the HMIWeb Object Model. The top level is the Station object, which contains Station-wide settings, such as title bar format, timers, and so on. From Station object you can access the RuntimeStatus and StationWindows properties. The RuntimeStatus object contains the status of the connection, and the StationWindows object contains an Application object for each Station window. For example, if you have a multi-window Station with four windows, there will be four instances of the Application object. If you have a single-window Station, there is only one instance of the Application object.



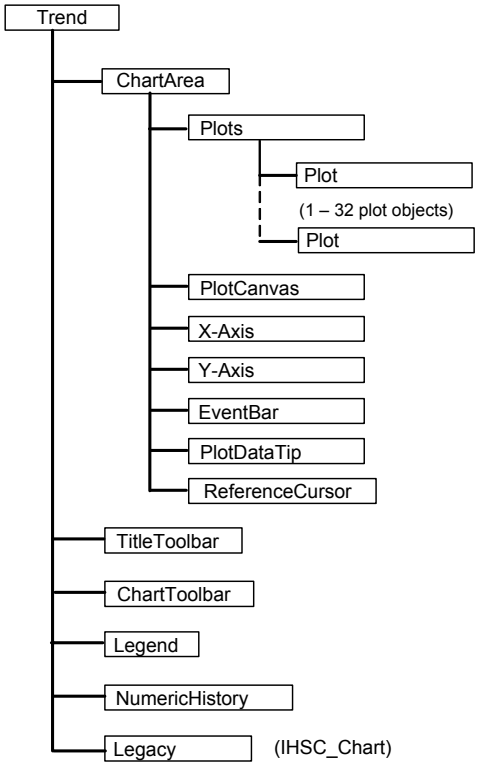
The following figure shows a simplified version of the DOM as it applies to displays. The main objects are the document (the display itself) and the children (the objects in the display).


Station
StationWindows
RuntimeStatus
Application
window
window.external
AppWindow
StationDataRepository
document
event
children
all
Item(..)
Item(..)
element

Overview of the trend object model

The following figure shows the levels of the trend object model. The top level is the Trend object, which provides methods and properties for controlling and manipulating a trend. It also provides access to a number of child objects which provide a greater level of control of the trend.

One of the child objects is the Legacy object which exposes the chart object for backward compatibility. (The chart object was supported in previous versions of the HMIWeb Display Builder.) If you have a display currently using the chart object, you need to migrate this display.



Objects

The objects are listed in alphabetical order.

ActiveX controls

Attention

Because of the extremely varied nature of ActiveX controls, you should thoroughly test the operation of displays that include ActiveX controls before releasing them to users.

Notes

- The effects of inserting an ActiveX control vary widely, according to the nature of the control. Some controls may, for example, initiate downloading and installation of control-related components.
- If any security-related warning appears, check that it is appropriate to the control you are inserting before proceeding. If you are unsure, you should refer to the relevant Microsoft documentation.
- Some controls add specialized properties to the Properties Window, which you must define. (See the documentation/help supplied with the control for details.)

Alphanumeric object

Properties

Property	Description
`currentValue`	Returns the current value of the object as last set by the server or script.

`Data*`
`Quality*`
`AlarmSeverity`
	These properties only exist if one or more point parameters are listed on the object's Script Data tab. See "Properties and methods of points on the Script Data tab" on page 306.

`disabled`	Returns or sets operator control of the object, the value is either `True` or `False`.
`fillColor`	Returns or sets the object's fill color. See "Setting and returning colors" on page 303.
`fillColorBlink`	Returns or sets the blinking state of the object's fill color. The value is either `True` or `False`.
`id`	Returns the object's name.
`lineColor`	Returns or sets the object's line color. See "Setting and returning colors" on page 303.
`lineColorBlink`	Returns or sets the blinking state of the object's line color. The value is either `True` or `False`.
`parentElement`	Returns the object's parent, which is either the group (if it is part of a group) or the page.
`style`	Returns or sets the object's style-related properties. See "style object" on page 407.
`styleClass`	Returns or sets the object's style. See "About style sheets" on page 108.
`textColor`	Returns or sets the object's text color. The color can be either the RGB value (24-bit) or one of the predefined "Setting and returning colors" on page 303.
`textColorBlink`	Returns or sets the blinking state of the object's text color. The value is either `True` or `False`.
`title`	Returns or sets the object's title (ToolTip). To create a multi-line ToolTip, insert `Chr(10)` between each line, for example: `"Line1" & Chr(10) & "Line2"`

Property	Description
value	Returns or sets the currently displayed value. **Note:** If you set the value of an alphanumeric linked to the database, the value will not change until the server has acknowledged the control. Performing large numbers of such controls where timing may be a critical factor is not recommended.

Methods

blur method

CancelChange method

CancelEvent method

click method

focus method

GetStyleProperty method

Events

onactivate event

onblur event

onchange event

onclick event

oncontextmenu event

ondatachange event

ondblclick event

ondeactivate event

onfocus event

onkeydown event

onkeypress event

onkeyup event

onmousedown event

onmouseenter event

onmouseleave event

onmousemove event

onmouseout event

onmouseover event

onmouseup event

onupdate event

Application object

Description

Each Application object represents a Station window (display). For a single-window Station, there is only one instance of the Application object. For multi-window Station, there are up to 20 instances of the Application object. The collection of Application objects are children of the StationWindows object.

Objects within each Station window can be accessed from the Application object instance for that window.

Attention

Page-level scripts (scripts in the current display) access the Application object of the display running the script using `window.external`. To access the Application object of other displays, you use the `Parent` property to gain access to the Station object. For example, to access the Application object of the display that has the input focus, you use `window.external.Parent.ActiveWindow`. See "Accessing Station-related properties in page-level scripts" on page 300.

Properties

Property	Description
`AlarmZonePointID`	Returns the ID of the current alarm point. The associated alarm appears in the Alarm Line. **Note:** For compatibility with earlier releases, this property is available from this object. For future developments, you should access this property from the RuntimeStatus object.
`AlarmZonePointServerAliasName`	Returns the alias of the server on which the current alarm point resides. **Note:** For compatibility with earlier releases, this property is available from this object. For future developments, you should access this property from the RuntimeStatus object.
`AlarmZonePointServerNetworkName`	Returns the basename of the server on which the current alarm point resides. **Note:** For compatibility with earlier releases, this property is available from this object. For future developments, you should access this property from the RuntimeStatus object.
`Application`	Returns the Application object.
`AppWindow`	Returns the AppWindow object.

Property	Description
AreSafeBrowseRightMouseClicksAllowed	Returns True if right-clicking a SafeBrowse object is allowed (that is, displays the shortcut menu). If not, it returns False.
CommandZoneText	Returns or sets the command that appears in the Command Zone. When used to set a command, the user must press **ENTER** to execute the command. **Note:** For compatibility with earlier releases, this property is available from this object. For future developments, you should access this property from the Station object.
ConnectionFile	Returns the filename (and path) of the Station setup file (*.stn) used for the current connection to the server. **Note:** For compatibility with earlier releases, this property is available from this object. For future developments, you should access this property from the Station object.
CurrentFieldOffset	Returns or sets the sequence number of the current field in the current record/database. For example, a value of 3 represents the third field in the record. Used in conjunction with the CurrentRecordOffset and CurrentFileOffset properties to uniquely identify a particular field.
CurrentFileOffset	Returns or sets the sequence number of the current file (in the set of files). For example, a value of 3 represents the third file in the set. Used in conjunction with the CurrentRecordOffset and CurrentFieldOffset properties to uniquely identify a particular field.
CurrentPage	Returns or sets the name of the current display. If set, Station displays that display.
CurrentPageLocation	Returns the full path/filename of the current display.
CurrentPointID	Returns the ID of the currently selected point.
CurrentPointServerAliasName	Returns the alias of the server on which the currently selected point resides.
CurrentPointServerNetworkName	Returns the basename of the server on which the currently selected point resides.

Property	Description
CurrentRecordOffset	Returns or sets the sequence number of the current record in the current file. For example, a value of 3 represents the third record. Used in conjunction with the CurrentFieldOffset and CurrentFileOffset to uniquely identify a particular field.
DataDeliveryTime	Returns the duration, in milliseconds, for Station to deliver the initial data updates to the current Station display.
DetailPointID	Only applicable if the current display is a Point Detail display. Returns the ID of the point whose details are currently displayed.
DetailPointServerAliasName	Only applicable if the current display is a Point Detail display. Returns the alias of the server on which the point, whose details are currently displayed, resides.
DetailPointServerNetworkName	Only applicable if the current display is a Point Detail display. Returns the base network name of the server on which the point, whose details are currently displayed, resides.
Dictionary	Returns the Dictionary object.
Document	Returns the pointer to the current display/page. If the current display is a DSP display, returns a NULL pointer and results in an error.
Environment	Returns the application environment type. • 0 = Station • 1 = Browser • 2 = Studio
LastSelectedPointID	Returns the point ID of the last selected point. **Note:** For compatibility with earlier releases, this property is available from this object. For future developments, you should access this property from the RuntimeStatus object.
MessageZoneText	Returns or sets the text that appears in the Station's Message Zone. **Note:** For compatibility with earlier releases, this property is available from this object. For future developments, you should access this property from the Station object.
OnPageCompleteTime	Returns the duration, in milliseconds, for Station to execute any OnPageComplete scripts in the current Station display.

Property	Description
OnUpdateTime	Returns the duration, in milliseconds, for Station to execute any OnUpdate scripts when loading the current Station display.
OperatorID	Returns the: • Operator ID if the Station is configured for operator-based security • Station number if the Station is configured for Station-based security **Note:** For compatibility with earlier releases, this property is available from this object. For future developments, you should access this property from the RuntimeStatus object.
Page	Returns the Page object. See page 380.
PageRenderTime	Returns the duration, in milliseconds, for Station to render the current Station display. This duration is from when Station starts drawing the display until the display is fully visible.
PageLoadTime	Returns the duration, in milliseconds, for Station to load and render the current Station display. This duration is from the user action to call up the display until the display is fully visible.
Parent	Returns the Station object.
parentElement	Returns the name of the element that launched a popup or faceplate within a multi-window Station display. **Note:** This property is not available within a single-window Station.
SecurityLevel	Returns the user's security level. The values are: • 0 = View Only • 1 = Ack Only • 2 = Operator • 3 = Supervisor • 4 = Engineer • 5 = Manager **Note:** For compatibility with earlier releases, this property is available from this object. For future developments, you should access this property from the RuntimeStatus object.

Property	Description
SecurityType	Returns the Station security type for the current connection. The values can either be: • 0 = Station-based security • 1 = Operator-based security • 2 = Operator-based security (Windows users allowed) • 3 = Operator-based security (Windows users allowed, Signon Manager enabled) • 4 = Operator-based security (Windows users allowed, single signon) **Note:** For compatibility with earlier releases, this property is available from this object. For future developments, you should access this property from the RuntimeStatus object.
ServerName	Returns the name of the server to which the Station is connected. If the Station is connected to a Console Station, returns the name of the Console Station. **Note:** For compatibility with earlier releases, this property is available from this object. For future developments, you should access this property from the RuntimeStatus object.
ServerVersion	Returns the version number of the server to which the Station is connected. **Note:** For compatibility with earlier releases, this property is available from this object. For future developments, you should access this property from the RuntimeStatus object.
StationDataRepository	Returns the StationDataRepository object.
StationNumber	Returns the Station's server connection number. **Note:** For compatibility with earlier releases, this property is available from this object. For future developments, you should access this property from the RuntimeStatus object.
Status	Returns the RuntimeStatus object.

Property	Description
`TemporaryMessageZoneText`	Sets the text of a temporary message that appears in the Station's Message Zone for 5 seconds (after which it is cleared). **Note:** If you set this property while there is a normal message in the Message Zone (represented by the MessageZoneText property), the temporary message will replace the normal one. Conversely, if a normal message is generated while the temporary message is in the Message Zone, the normal message will replace the temporary one. **Note:** For compatibility with earlier releases, this property is available from this object. For future developments, you should access this property from the Station object.
`window`	Returns the pointer to the HTML Window, which gives access to the display's window-related properties. If the current display is a DSP display, returns a NULL pointer and results in an error.
`WindowMode`	Returns Station's window mode, which can be: • `0` = Single window • `1` = Multi-window **Note:** For compatibility with earlier releases, this property is available from this object. For future developments, you should access this property from the Station object.

Remarks

- Events fired by the Application object can only be handled by Station scripts or Station Scripting Objects (ActiveX controls, loaded by Station at runtime, that hook into Station's application-level events). See "Writing Station-level scripts" on page 320.

Methods

Blue method

CancelOperatorCommand method

CancelResponse method

Connect method

DisplayExists method

ExecuteOperatorCommand method

FindFile method

FlagUserInteraction method

GetConnectionProperty method

GetResponse method

GetResponse2 method

GetResponse3 method

GetResponse4 method

Green method

InvokeCommand method

InvokeMenu2 method

InvokePopup method

KillTimer method

LogMessage method

MakeColor method

Quit method

Red method

Refresh method

RequestServerLRN method

RequestTask method

SendKeyPress method

Shell method

UserObjectNotify method

Events

OnActivated event

OnAlarm event

OnAppStartup event

OnAppStatusUpdate event

OnBeforeConnect event

OnBeforeOKBKeyPressed event

OnBlink event

OnConnect event

OnDictionaryValueChanged event

OnDisconnect event

OnDisplayNotFound event

OnMenu event

OnMenu2 event

OnOKBKeyPressed event

OnOperatorChange event

OnOperatorCommand event

OnOperatorCommand2 event

OnPageLoad event

OnPeriodicUpdate event

OnQuit event

OnResponse event

OnSilenceIKB event

OnTimer event

OnUserObjectNotify event

Example

An external application would initially use code similar to the following to create a reference to Station. (The CreateObject function is a standard VBScript function.)

```
Set objStationApp = CreateObject("Station.Application")
```

The external application can then control Station through the object variable `objStationApp`. For example, to instruct Station to call up the Alarm Summary (whose display number is 5), the application would use the following code.

```
objStationApp.CurrentPage = 5
```

This is example is for controlling one instance of Station only.

Example scenarios

Controlling Station from another application

Responding to Station events from another application

AppWindow object

Description

The AppWindow object represents the Station window, and contains the properties that reflect the current state of the main window.

It is exposed as a property of the Application object, which is Station's root object.

Attention

Page-level scripts (scripts in the current display) access the AppWindow object using `window.external.AppWindow`. See "Accessing Station-related properties in page-level scripts" on page 300.

Properties

Property	Description
`Application`	Returns the Application object.
`DisplayHWND`	Returns the "handle" allocated by Windows to the display window.
`Height`	Returns or sets the height of the Station window. The range depends on Windows' screen setting. For example, if the setting is 1280x1024, the height can be up to 1024 pixels.
`HWND`	Returns the "handle" allocated by Windows to the Station window.
`Left`	Returns or sets the position of the left edge of the Station window with reference to the left edge of the screen. The range depends on the Windows screen setting.
`MenuVisible`	Sets the visibility of Station's Menu. The values is either `True` (visible) or `False` (hidden).
`Parent`	Returns the Application object.
`Title`	Returns the title that appears in the title bar of the Station window.
`Top`	Returns or sets the position of the top of the Station window with reference to the top of the screen. The range depends on the Windows screen setting.
`Width`	Returns or sets the width of the Station window. The range depends on the Windows screen setting. For example, if the screen has a 1280x1024 resolution, the width can be up to 1280.

Property	Description
`WindowState`	Returns or sets the state of the Station window. The values are: • `0` = Window is hidden (not visible). Note that this value should be used with caution because a hidden Station will not receive any user events, although it will continue to receive server and timer events. • `1` = Window is visible and can be sized by the user • `2` = Window is minimized • `3` = Window is maximized

Methods

DPToLPX and DPToLPY methods

LPToDPX and LPToDPY methods

Attention

For compatibility with earlier releases, the remaining methods are available from this object. For future developments, you should access these methods from the Station object.

MessageBox method

SetCommandUIState method

SetTitleBarFormat method

Events

None.

ChartArea object

Description

The ChartArea object is a child of the Trend object. It includes the plot canvas where the plots are drawn as well as the X and Y axes.

It provides methods and properties for manipulating and control the trend's chart area.

For more information about the Trend object model, see "Overview of the trend object model" on page 336.

Properties

Property	Description
`BackColor`	Returns or sets the background color.
`BarGraph`	Returns or sets whether the chart is display a bar graph for the currently selected plot.
`BorderVisible`	Returns or sets the visibility of the border, which can be: • `0` = Invisible • `1` = Visible
`EventBar`	Returns the EventBar object
`Pause`	Returns or sets the paused state of the trend.
`PercentMode`	Returns or sets the units of the Y-axis to percent or engineering units.
`PlotCanvas`	Returns the PlotCanvas object
`PlotDataTip`	Returns the PlotDataTip object
`Plots`	Returns the Plots object.
`ReferenceCursor`	Returns the ReferenceCursor object
`XAxis`	Returns the XAxis object
`XAxisToolbarVisible`	Returns or sets the visibility of the X axis toolbar which can be: • `0` = Invisible • `1` = Visible The X axis toolbar contains buttons for pause, resume and zooming actions.
`YAxis`	Returns the YAxis object

Methods

GetXAxisToolbarItemVisible method

SetXAxisToolbarItemVisible method

Events

None.

ChartToolbar object

Description

The ChartToolbar object is a child object of the Trend object.

For more information about the Trend object model, see “Overview of the trend object model” on page 336.

Properties

Property	Description
`visible`	Sets the visibility of the chart toolbar.

Methods

GetItemVisible method

SetItemVisible method

Events

None.

Checkbox object

Properties

Property	Description
`currentValue`	Returns the current value of the object as last set by the server or script.

`Data*`
`Quality*`
`AlarmSeverity`
	These properties only exist if one or more point parameters are listed on the object's Script Data tab. See "Properties and methods of points on the Script Data tab" on page 306.

`disabled`	Returns or sets operator control of the object, the value is either `True` or `False`.
`fillColor`	Returns or sets the object's fill color. See "Setting and returning colors" on page 303.
`fillColorBlink`	Returns or sets the blinking state of the object's fill color. The value is either `True` or `False`.
`id`	Returns the object's name.
`label`	Returns or sets the checkbox's label.
`lineColor`	Returns or sets the object's line color. See "Setting and returning colors" on page 303.
`lineColorBlink`	Returns or sets the blinking state of the object's line color. The value is either `True` or `False`.
`parentElement`	Returns the object's parent, which is either the group (if it is part of a group) or the page.
`style`	Returns or sets the object's style-related properties. See "style object" on page 407.
`styleClass`	Returns or sets the object's style. See "About style sheets" on page 108.
`textColor`	Returns or sets the object's text color. The color can be either the RGB value (24-bit) or one of the predefined "Setting and returning colors" on page 303.
`textColorBlink`	Returns or sets the blinking state of the object's text color. The value is either `True` or `False`.
`title`	Returns or sets the object's title (ToolTip). To create a multi-line ToolTip, insert `Chr(10)` between each line, for example: `"Line1" & Chr(10) & "Line2"`
`value`	Returns or sets the value, which can be: • `0` = Unchecked • `1` = Checked

Methods

blur method

CancelChange method

CancelEvent method

click method

focus method

GetStyleProperty method

Events

onactivate event

onblur event

onchange event

onclick event

oncontextmenu event

ondatachange event

ondblclick event

ondeactivate event

onfocus event

onkeydown event

onkeypress event

onkeyup event

onmousedown event

onmouseenter event

onmouseleave event

onmousemove event

onmouseout event

onmouseover event

onmouseup event

onupdate event

Combobox object

Properties

Property	Description
`count`	Returns the number of items in the object's drop-down list.
`currentValue`	Returns the current value of the object as last set by the server or script.

`Data*`
`Quality*`
`AlarmSeverity`
	These properties only exist if one or more point parameters are listed on the object's Script Data tab. See "Properties and methods of points on the Script Data tab" on page 306.

`disabled`	Returns or sets operator control of the object, the value is either `True` or `False`.
`fillColor`	Returns or sets the object's fill color. See "Setting and returning colors" on page 303.
`fillColorBlink`	Returns or sets the blinking state of the object's fill color. The value is either `True` or `False`.
`id`	Returns the object's name.
`label`	Returns or sets the checkbox's label.
`lineColor`	Returns or sets the object's line color. See "Setting and returning colors" on page 303.
`lineColorBlink`	Returns or sets the blinking state of the object's line color. The value is either `True` or `False`.
`parentElement`	Returns the object's parent, which is either the group (if it is part of a group) or the page.
`selectedIndex`	Returns or sets the index that identifies an item in the drop-down list. The value for the first item is 0, the value for the second item is 1, and so on. A value of -1 indicates that no item is selected.
`style`	Returns or sets the object's style-related properties. See "style object" on page 407.
`styleClass`	Returns or sets the object's style. See "About style sheets" on page 108.
`textColor`	Returns or sets the object's text color. The color can be either the RGB value (24-bit) or one of the predefined "Setting and returning colors" on page 303.
`textColorBlink`	Returns or sets the blinking state of the object's text color. The value is either `True` or `False`.

Property	Description
`title`	Returns or sets the object's title (ToolTip). To create a multi-line ToolTip, insert `Chr(10)` between each line, for example: `"Line1" & Chr(10) & "Line2"`
`value`	Returns or sets the value (a string) displayed in the combobox.

Methods

AddString method

DeleteString method

ResetContent method

CancelChange method

CancelEvent method

GetStyleProperty method

Events

onactivate event

onblur event

onchange event

onclick event

oncontextmenu event

ondatachange event

ondblclick event

ondeactivate event

onfocus event

onkeydown event

onkeypress event

onkeyup event

onmousedown event

onmouseenter event

onmouseleave event

onmousemove event

onmouseout event

onmouseover event

onmouseup event

onupdate event

Dictionary object

Description

The Dictionary is a collection of named variables (items), which can be used by any script.

Properties

Property	Description
`Application`	Returns the Application object.
`Count`	Returns the number of items in the collection.
`Parent`	Returns the Application object.

Dictionary Item Properties

Property	Description
`Name`	Returns the item's name. The name is case-sensitive.
`Value`	Returns or sets the item's value. The value can be of any form, such as text or numeric.

Remarks

- By default, items remain in the Dictionary for the life of Station. It is therefore important to use the Remove method to remove items if they are no longer required.

Methods

Add method

Item method

Remove method

Events

None.

Example

This example shows how to create a variable called "Volts" and assign it a value:

```
Window.External.Dictionary.Add "Volts", 12.2
```

Once created, any script can access the value as follows:

```
BattVolts = Window.External.Dictionary.Item("Volts").Value
```

DisplayDataRepository object

Description

The DisplayDataRepository object is a collection of named system custom properties with the scope of display, which can be used by any script on a page.

Properties

Property	Description
`Count`	Returns the number of properties in the repository
`autoReloadContent`	Returns or sets automatic reloading of the custom property when there is a change in the value. The values are: • `True` = Automatic reload is enabled (default) • `False` = Automatic reloading is disabled

Remarks

- To improve the performance if you are changing the value of a lot of properties, you should set this back to `True`.

Methods

Add method

GetValue method

PutValue method

Example

This example shows how to add the system custom property called “Custom1” and assign it as a point type and assign the default value as “sinewave”.

```
DisplayDataRepository.Add(“Custom1”, “Point”, “sinewave”)
```

This example gets the value of the custom property named “Custom1”.

```
DisplayDataRepository.GetValue(“Custom1”)
```

This example sets the value of the custom property “Custom1” to “tank1”.

```
DisplayDataRepository.PutValue(“Custom1” “tank1”)
```

event object

Description

The event object is a standard DHTML object that represents the state of an event, such as the object for which the event occurred, the state of the keyboard keys, the location of the mouse and the state of the mouse buttons.

For a detailed description of the event object, go to Microsoft's MSDN Web site at `http://msdn.microsoft.com/library/default.asp?url=/workshop/author/dhtml/reference/objects/obj_event.asp`

Properties

The following table only describes properties commonly used in display scripts.

Property	Description
`cancelBubble`	Returns or sets event bubbling (which determines whether the current event bubbles up the hierarchy of event handlers). The value is either `True` or `False`.
`keyCode`	Returns or sets the key code associated with the key that caused the event.
`propertyName`	Returns or sets the name of the property that changes on the object.
`returnValue`	Returns or sets event the return value from the event. The value can be either: • `True` (default) = Value from the event is returned. • `False` = Default action of the event on the source object is canceled.
`srcElement`	Returns or sets the object that fired the event.
`type`	Returns or sets the event name (without the "on" prefix) from the event object.

Remarks

- The event object is only available during an event—that is, you can use it in event handlers but not in other code.
- Although all event properties are available to all event objects, some properties might not have meaningful values during some events.

Example

This example uses the event object and its srcElement property to implicitly reference the object that caused the event. In this example, the object's fill color is changed to red.

```
window.event.srcElement.fillColor = vbRed
```

This example uses the event object and its returnValue property to cancel the an object's default action. (For example, it could prevent the default action of a function key on an IKB being performed when the user presses the key.)

```
window.event.srcElement.returnValue = False
```

EventBar object

Description

The EventBar object is a child object of the Trend object. It contains properties to control and manipulate the event bar in the chart area when the trend has events visible.

For more information about the Trend object model, see “Overview of the trend object model” on page 336.

Properties

Property	Description
`BackgroundColor`	Returns or sets the background color of the event bar.
`BorderStyle`	Returns or sets the border style for the axis, which can be: • `0` = No border • `1` = Flat border (the default) • `2` = Sunken border • `3` = Raised border • `4` = 3D bar border
`BorderVisible`	Returns or sets the visibility of the event bar border.
`BorderWidth`	Returns or sets the width, in pixels, of the event bar border.
`DataTipBackColor`	Returns or sets the background color of the event data tips that appear when an event symbol is clicked.
`DataTipBorderColor`	Returns or sets the color of the data tip border.
`Visible`	Returns or sets the visibility of the event bar.

Methods

GetDataTipArrowColors method

GetDataTipVisible method

HideAllDataTips method

HideDataTips method

SetDataTipArrowColors method

SetDataTipVisible method

Events

None.

Group object

Properties

Property	Description
`all` `children`	These are standard DHTML collections that return a reference to the collection of child objects. To learn how to use them, see “Referencing child objects with the children and all collections” on page 294.

`Data*`
`Quality*`
`AlarmSeverity`
	These properties only exist if one or more point parameters are listed on the object’s Script Data tab. See “Properties and methods of points on the Script Data tab” on page 306.

`id`	Returns the object’s name.
`parentElement`	Returns the object’s parent, which is either the group (if it is part of a group) or the page.
`Rotation`	Returns or sets the clockwise angle (degrees) through which the object is rotated.
`style`	Returns or sets the following style-related properties: • Position • Cursor • Visibility Other properties typically set by the style object are specified using object-specific properties. See “style object” on page 407.
`styleClass`	Returns or sets the object’s style. See “About style sheets” on page 108.
`title`	Returns or sets the object’s title (ToolTip). To create a multi-line ToolTip, insert `Chr(10)` between each line, for example: `"Line1" & Chr(10) & "Line2"`

Methods

GetStyleProperty method

Events

onactivate event

onblur event

onclick event

oncontextmenu event

ondatachange event
ondblclick event
ondeactivate event
onfocus event
onkeydown event
onkeypress event
onkeyup event
onmousedown event
onmouseenter event
onmouseleave event
onmousemove event
onmouseout event
onmouseover event
onmouseup event
onupdate event

Hyperlink object

Properties

Property	Description

`Data*`
`Quality*`
`AlarmSeverity`
	These properties only exist if one or more point parameters are listed on the object's Script Data tab. See "Properties and methods of points on the Script Data tab" on page 306.

`disabled`	Returns or sets operator control of the object, the value is either `True` or `False`.
`HREF`	Returns or sets the URL. To set the URL, see "Specifying URLs" on page 221.
`id`	Returns the object's name.
`innerText`	Returns or sets the object's text.
`parentElement`	Returns the object's parent, which is either the group (if it is part of a group) or the page.
`style`	Returns or sets the object's style-related properties. See "style object" on page 407.
`styleClass`	Returns or sets the object's style. See "About style sheets" on page 108.
`title`	Returns or sets the object's title (ToolTip). To create a multi-line ToolTip, insert `Chr(10)` between each line, for example: `"Line1" & Chr(10) & "Line2"`

Methods

blur method

click method

focus method

GetStyleProperty method

Events

onactivate event

onblur event

onclick event

oncontextmenu event

ondatachange event

ondblclick event
ondeactivate event
onfocus event
onkeydown event
onkeypress event
onkeyup event
onmousedown event
onmouseenter event
onmouseleave event
onmousemove event
onmouseout event
onmouseover event
onmouseup event

Indicator object

Properties

Property	Description

`Data*`
`Quality*`
`AlarmSeverity`
	These properties only exist if one or more point parameters are listed on the object's Script Data tab. See "Properties and methods of points on the Script Data tab" on page 306.

`disabled`	Returns or sets operator control of the object, the value is either `True` or `False`.
`id`	Returns the object's name.
`fillColor`	Returns or sets the object's fill color. See "Setting and returning colors" on page 303.
`fillColorBlink`	Returns or sets the blinking state of the object's fill color. The value is either `True` or `False`.
`levelFillColor`	Returns or sets the object's level fill color. The color can be either the RGB value (24-bit) or one of the predefined "Setting and returning colors" on page 303.
`lineColor`	Returns or sets the object's line color. See "Setting and returning colors" on page 303.
`lineColorBlink`	Returns or sets the blinking state of the object's line color. The value is either `True` or `False`.
`parentElement`	Returns the object's parent, which is either the group (if it is part of a group) or the page.
`rangeHi` `rangeLo`	Return or set the upper and lower range of the indicator. If they aren't specified they are considered to be `0` to `100`.
`rangeOrigin`	Returns or sets the fill starting point of the indicator as a percentage of the actual range. The value can be less than `0` and greater than `100`.
`style`	Returns or sets the object's style-related properties. See "style object" on page 407.
`styleClass`	Returns or sets the object's style. See "About style sheets" on page 108.
`title`	Returns or sets the object's title (ToolTip). To create a multi-line ToolTip, insert `Chr(10)` between each line, for example: `"Line1" & Chr(10) & "Line2"`
`value`	Returns or sets the currently displayed value (the level shown by the indicator).

Property	Description
visibleRangeHi visibleRangeLo	Returns or sets the upper and lower bound of values displayed by the indicator. They are constant values that are a percentage of the actual range. The values can be greater than 100 and less than 0.

Methods

blur method

focus method

GetStyleProperty method

Events

onactivate event

onblur event

onclick event

oncontextmenu event

ondatachange event

ondblclick event

ondeactivate event

onfocus event

onkeydown event

onkeypress event

onkeyup event

onmousedown event

onmouseenter event

onmouseleave event

onmousemove event

onmouseout event

onmouseover event

onmouseup event

onupdate event

Legacy object

Description

This object replaces the Chart object and is supported for backward compatibility only. To create or manipulate a trend through scripting, use the Trend object and it's child objects.

For more information about the Trend object model, see "Overview of the trend object model" on page 336.

Properties

Property	Description
`axisBackColor`	Returns or sets the axis background color.
`AxisLabelColor`	Returns or sets the color of the axis labels.
`AxisMarkerLabelColor`	Returns or sets the text color of the axis marker labels.
`chartType`	Returns or sets the chart's type, which can be: • `0` = Line chart • `1` = Bar chart
`CurrentPlot`	Returns or sets the plot ID of the currently selected plot. The plot ID is a number assigned by the chart when the plot is added to it. (The plot ID is returned by a call to the AddNewPlot method.) A new chart always contains one plot—called the "default" plot—whose plot ID is always `0`. (Note that it cannot be assumed that the next plot to be added will have a plot ID of 1.)
`dataTipsEnable`	Returns or sets the data tips state. If set to `True` (the default) then data tips (tool tips on the plot) are enabled.
`dataTracking`	Returns or sets data tracking/autoscrolling. If set to `True` (the default), then when data updates are received the chart automatically scrolls them into view.
`plotBackColor`	Returns or sets the plot background color.
`referenceLineColor`	Returns or sets the reference line color. See "Setting and returning colors" on page 303.
`scrollBarsVisible`	Returns or sets the visibility of the scrollbars, which can be: • `0` = Invisible • `1` = Visible

Remarks

- Unlike most other objects documented in this help, the chart object is an ActiveX control. (Most objects, such as the Alphanumeric and Indicator, are native HMIWeb objects.)
- The chart object does not support standard DHTML methods or events (such as click, blur, onblur and onclick). It only supports the methods and events listed below.

Methods

AddNewPlot method

AddPlotData method

ClearPlotData method

Copy method

GetPlotColor method

GetPlotDiscrete method

GetPlotMarkerStyle method

GetPlotVisible method

GetReferenceValues method

GetStyleProperty method

Redraw method

RemovePlot method

RemovePlotData method

SetPlotColor method

SetPlotDiscrete method

SetPlotMarkerStyle method

SetPlotPen method

SetPlotRange method

SetPlotStepMode method

SetPlotVisible method

UnZoom method

Events

Legacy_OnChartLoadComplete event

Legacy_OnChartUnZoomed event

Legacy_OnChartScroll event

Legacy_OnChartZoom event

Legacy_OnReferenceLineSet event

Legend object

Description

The Legend object is a child object of the Legacy object. It contains properties to control and manipulate the legend associated with the trend's chart.

For more information about the Trend object model, see "Overview of the trend object model" on page 336.

Properties

Property	Description
CurrentPlot	Returns or sets the plot ID of the currently selected plot.
CurrentValue	Returns the current value of the plot in the given row.
Description	Returns the description of the plot in the given row.
GridBackColor	Returns or sets the background color of the legend grid.
GridFontName	Returns or sets the font of the legend grid.
GridFontSize	Returns or sets the font size of the legend grid.
GridForeColor	Returns or sets the foreground color of the legend grid.
HeaderBackColor	Returns or sets the background color of the legend header including column and row headers.
HeaderFontName	Returns or sets the font of the header.
HeaderFontSize	Returns or sets the font size of the header.
HeaderForeColor	Returns or sets the foreground color of the legend header including column and row headers.
HighRange	Returns the high range of the plot in the given row.
HighScale	Returns or sets the high scale of the plot in the given row.
LowRange	Returns the low range of the plot in the given row.
LowScale	Returns or sets the low scale of the plot in the given row.
Parameter	Returns or sets the point parameter of the plot in the given row.
PlotColor	Returns or sets the color of the plot in the given row.
PlotVisible	Returns or sets the visibility of the plot in the given row. The values are
PointID	Returns or sets the point ID of the plot in the given row.
ReferenceValue	Returns the reference value of the plot in the given row.
Unit	Returns the unit of the plot in the given row.

Property	Description
`visible`	Returns or sets the visibility of the legend in the trend. The values are

Methods

None.

Events

None.

Node object

Description

Represents a node of a Vector Graphic object.

Properties

Property	Description
x *y*	The x-and y-coordinates of the node.

Remarks

- The Nodes object represents all the object's nodes.
- To learn about vector graphic objects, see "About vector graphic objects" on page 222.

Methods

SetNodePoint method

Events

None.

Example

This example gets the x- and y-coordinates of the first node in a polygon.

```
nX = polygon001.Nodes.Item(0).x
nY = polygon001.Nodes.Item(0).y
```

Nodes object

Description

Nodes is a collection that represents all the nodes of a Vector Graphic object.

Properties

Property	Description
`Count`	Returns the number of nodes in the collection.
`Item(`*`index`*`)`	Returns the Node object with the specified *`index`*. The index is an integer, and index of the first node in the collection is `0`.

Remarks

- The Node object represents an individual node.
- An error will occur if you specify an index that doesn't exist.
- To learn about vector graphic objects, see "About vector graphic objects" on page 222.

Methods

None.

Events

None.

Example

This example returns the last node of a polygon.

```
dim nCount
dim objNode
nCount = polygon001.Nodes.Count
set objNode = polygon001.Nodes.Item(nCount-1)
```

NumericHistory object

Description

The NumericHistory object is a child object of the Trend object. It contains properties to control and manipulate the history table associated with a trend.

For more information about the Trend object model, see “Overview of the trend object model” on page 336.

Properties

Property	Description
`visible`	Returns or sets the visibility of the history table in the trend.

Methods

None.

Events

None.

Page object

Description

The Page object represents the current display (or shape, popup, faceplate). It contains all the objects in the display, which can be referenced through its "all" collection, and is also used for firing events related to the lifetime of the page, such as onpagecomplete.

Properties

Property	Description
`all` `children`	These are standard DHTML collections that return a reference to the collection of child objects. To learn how to use them, see "Referencing child objects with the children and all collections" on page 294.
`style`	Returns or sets the display's style-related properties. See the style object.
`styleClass`	Returns or sets the display's background style.
`stylesheetSrc`	Returns or sets the URL of the style sheet attached to the display. See "About style sheets" on page 108.
`ScriptErrorHandler`	Specifies the name of the current error handler. It is set to NULL when the display is loaded or refreshed, and can be dynamically altered by scripts. See "Using error handlers" on page 323.

Remarks

- You reference the Page object using "page". This example obtains the URL of the style sheet currently attached to the display.

  ```
  x = page.stylesheetSrc
  ```

- If it is a faceplate the OnMenu2 event and OnOperatorKey event will not fire unless the faceplate has focus.

Methods

GetStyleProperty method

GetStyleClassProperty method

Events

onactivate event

OnActivated event

OnAlarm event

OnAppStatusUpdate event
OnBeforeConnect event
OnBlink event
onblur event
onclick event
OnConnect event
oncontextmenu event
ondblclick event
ondeactivate event
OnDictionaryValueChanged event
OnDisconnect event
onfocus event
onkeydown event
onkeypress event
onkeyup event
OnMenu2 event
onmousedown event
onmouseenter event
onmouseleave event
onmousemove event
onmouseout event
onmouseover event
onmouseup event
OnOperatorChange event
OnOperatorCommand2 event
onoperatorkey event
onpagecomplete event
OnPeriodicUpdate event
OnResponse event
onunload event
OnUserObjectNotify event

Picture object

Properties

Property	Description

`Data*`
`Quality*`
`AlarmSeverity`
	These properties only exist if one or more point parameters are listed on the object's Script Data tab. See "Properties and methods of points on the Script Data tab" on page 306.

`id`	Returns the object's name.
`parentElement`	Returns the object's parent, which is either the group (if it is part of a group) or the page.
`Rotation`	Returns or sets the clockwise angle (degrees) through which the object is rotated.
`src`	Returns or sets the name of the currently displayed picture file.
`style`	Returns or sets the object's style-related properties. See "style object" on page 407.
`styleClass`	Returns or sets the object's style. See "About style sheets" on page 108.
`title`	Returns or sets the object's title (ToolTip). To create a multi-line ToolTip, insert `Chr(10)` between each line, for example: `"Line1" & Chr(10) & "Line2"`

Methods

GetStyleProperty method

Events

onactivate event

onblur event

onclick event

oncontextmenu event

ondatachange event

ondblclick event

ondeactivate event

onfocus event

onkeydown event

onkeypress event

onkeyup event

onmousedown event

onmouseenter event

onmouseleave event

onmousemove event

onmouseout event

onmouseover event

onmouseup event

Plot object

Description

The Plot object allows for manipulation and control of the specific plot. There can be up to 32 Plot objects, one for each plot in the trend.

The Plot object is a child of the Plots object.

For more information about the Trend object model, see “Overview of the trend object model” on page 336.

Properties

Property	Description
`Color`	Returns or sets the color of the specified plot.
`DiscreteMode`	Returns or sets the visibility of the line for the specified plot. If set to true, then the markers are visible at each sample and no line is visible.
`MarkerStyle`	Returns or sets the marker used to draw the plot. If a marker is used, the mark is drawn at each sample point in the plot. Valid values are: • `1` = X • `2` = + • `3` = Square • `4` = Diamond • `5` = Triangle • `6` = Inverted triangle • `7` = Circle • `19` = Filled circle • `20` = Filled diamond • `21` = Filled triangle • `22` = Filled inverted triangle
`SampleCount`	Returns the number of samples currently stored for the plot.
`StepMode`	Returns or sets the step mode for the plot line. In step mode, the plot line is drawn horizontally to the next sample, where are vertical line then connects to the sample. In this mode, the plot looks like a series of steps.
`Visible`	Returns or sets the visibility of the specified plot.

Methods

Autoscale method

GetLabels method

GetPen method

GetYAxisScale method

GetYAxisUnzoomedScale

SetLabels method

SetPen method

SetYAxisScale method

SetYAxisUnzoomedScale method

Events

None.

Plots object

Description

The Plots object is a child object of the ChartArea object. It is a representation of a collection of Plot objects. You can access each individual plot in the trend from the Plots object including selecting the current plot, modifying the visible ranges of the plots, and adding or removing plots.

For more information about the Trend object model, see “Overview of the trend object model” on page 336.

Properties

Property	Description
`Count`	Returns the number of configured plots.
`Current`	Returns the plot ID of the currently selected plot.
`HighlightCurrentPlot`	Returns or sets highlighting for the currently selected plot. If true, the plot is drawn 3 pixels wide instead of 1 pixel.
`Item`	Returns the Plot object for the specified plot ID.
`PercentMode`	Returns or sets the units of the Y-axis to percent or engineering units.
`SingleScale`	Returns or sets a single scale for all plots.

Methods

Autoscale method

GetXAxisScale method

GetXAxisUnzoomedScale

GetYAxisScale method

GetYAxisUnzoomedScale

SetXAxisScale method

SetXAxisUnzoomedScale method

SetYAxisScale method

SetYAxisUnzoomedScale method

Events

None.

PlotCanvas object

Description

The PlotCanvas object is a child object of the ChartArea object. It contains properties for controlling and manipulating the visual aspects of the plot canvas. The plot canvas is the visual area within the chart area on which the plots are rendered.

For more information about the Trend object model, see “Overview of the trend object model” on page 336.

Properties

Property	Description
`BackgroundColor`	Returns or sets the background color on which the plots are drawn.
`BorderColor`	Returns or sets the color of the plot canvas border if the border is visible.
`BorderStyle`	Returns or sets the border style for the plot area within the chart.
`BorderVisible`	Returns or sets the visibility of the plot canvas border.
`BorderWidth`	Returns or sets the width of the plot canvas border, if the border is visible.
`EnableZooming`	Returns or sets operator control of zooming the chart.
`GridLineAutoColor`	Returns or sets automatic coloring of the gridline.
`GridLineStyle`	Returns or sets the line style for the grid line.
`GridLinesVisible`	Returns or sets the visibility of gridlines in the plot area.
`ZoomMode`	Returns or sets the zoom mode for the chart, which are: • `0` = Zoom X and Y axis • `1` = Zoom X axis only • `2` = Zoom Y axis only

Methods

DecreaseZoomLevel

GetGridLinePen method

GetZoomRegion method

IncreaseZoomLevel method

SetGridLinePen method

SetZoomRegion method

UnZoom method

Events

None.

PlotDataTip object

Description

The PlotDataTip object is a child object of the ChartArea object. It contains properties for controlling the plot data tips in the chart area.

For more information about the Trend object model, see “Overview of the trend object model” on page 336.

Properties

Property	Description
`DateFormat`	Returns or sets the format for any date/time information in the data tips on the chart. By default, the regional settings for the computer is used.
`Enable`	Returns or sets the data tips state. If set to `True`, then data tips (too tips on the plot) are enabled.
`Format`	Returns or sets the format of the data tips. The format specifies what information is displayed and the order of the information. • %p = Plot name • %y = Y-axis value • %Y = Y-axis units • %x = X-axis value • %X = X-axis units • %% = Add the percent text.
`UseCustomDataTips`	Returns or sets the source of data tips. If set to true, you provide your own data tips by handling data tip events and specifying the data tip you want displayed.

Methods

None.

Events

None.

Popup object

Properties

None.

Methods

CreatePopupWindow method

FindPopup method

Events

None.

Pushbutton object

Properties

Property	Description

`Data*`
`Quality*`
`AlarmSeverity`
	These properties only exist if one or more point parameters are listed on the object's Script Data tab. See "Properties and methods of points on the Script Data tab" on page 306.

`disabled`	Returns or sets operator control of the object. The value can be either `True` or `False`.
`fillColor`	Returns or sets the object's fill color. See "Setting and returning colors" on page 303.
`fillColorBlink`	Returns or sets the blinking state of the object's fill color. The value is either `True` or `False`.
`id`	Returns the object's name.
`lineColor`	Returns or sets the object's line color. See "Setting and returning colors" on page 303.
`lineColorBlink`	Returns or sets the blinking state of the object's line color. The value is either `True` or `False`.
`parentElement`	Returns the object's parent, which is either the group (if it is part of a group) or the page.
`style`	Returns or sets the object's style-related properties. See "style object" on page 407.
`styleClass`	Returns or sets the object's style. See "About style sheets" on page 108.
`textColor`	Returns or sets the object's text color. The color can be either the RGB value (24-bit) or one of the predefined "Setting and returning colors" on page 303.
`textColorBlink`	Returns or sets the blinking state of the object's text color. The value is either `True` or `False`.
`title`	Returns or sets the object's title (ToolTip). To create a multi-line ToolTip, insert `Chr(10)` between each line, for example: `"Line1" & Chr(10) & "Line2"`
`value`	The pushbutton's label.

Methods

blur method

CancelChange method

CancelEvent method

click method

focus method

GetStyleProperty method

Events

onactivate event

onblur event

onclick event

oncontextmenu event

ondatachange event

ondblclick event

ondeactivate event

onfocus event

onkeydown event

onkeypress event

onkeyup event

onmousedown event

onmouseenter event

onmouseleave event

onmousemove event

onmouseout event

onmouseup event

onmouseover event

onmouseup event

ReferenceCursor object

Description

The ReferenceCursor Object is a child object of the ChartArea object. It contains properties for controlling and manipulating the reference cursor in the chart area.

For more information about the Trend object model, see “Overview of the trend object model” on page 336.

Properties

Property	Description
`Color`	Returns or sets the color of the reference cursor
`Enable`	Returns or sets operator control of the reference cursor on the chart.
`SnapToValue`	Returns or sets the placement of the reference cursor. If set to `True`, the reference cursor is placed exactly where the mouse is clicked. All values corresponding to the reference cursor are interpolated between the nearest two real samples. If set to `False`, the reference cursor is placed at the nearest real sample.
`Visible`	Returns or sets the visibility of the reference cursor on the chart.
`X_Location`	Returns or sets the x-axis value of the reference cursor. This is typically a date.
`X_Rel_Position`	Returns or sets the relative position on the x-axis of the reference cursor. 0.0 = The left-most point of the visible range 1.0 = The right-most point of the visible range

Methods

GetDataTipFormat method

GetLineStyle method

GetReferenceValues method

SetDataTipFormat method

SetLineStyle method

Events

None.

RuntimeStatus object

Description

The RuntimeStatus object represents Station's status details.

It is exposed as a property of the Station object, which is Station's root object.

Attention

Page-level scripts (scripts in the current display) access the RuntimeStatus object using `window.external.Parent.RuntimeStatus`. See "Accessing Station-related properties in page-level scripts" on page 300.

Properties

Property	Description
`AlarmDescription`	Returns the alarm description that appears in the Alarm Zone.
`AlarmZonePointID`	Returns the ID of the current alarm point. (The associated alarm appears in the Alarm Line.)
`AlarmZonePointServerAliasName`	Returns the alias of the server on which the current alarm point resides.
`AlarmZonePointServerNetworkName`	Returns the basename of the server on which the current alarm point resides.
`ClusterServerLinkName`	Returns the name of the cluster server link if the Station is connected to a Console Station. This is used when the cluster server contains dual network cards and you need to determine which card on the cluster server Station is connected to.
`ClusterServerName`	Returns the name of the cluster server if the Station is connected to a Console Station.
`Date`	Returns the date that appears in the Status Line.

`FieldColor(`*`fieldid`*`)`
`FieldState(`*`fieldid`*`)`
`FieldText(`*`fieldid`*`)`
	Return the color, state and text of the specified box in the Status Line. For a description of identifier and possible statuses of each box, see "Status Line boxes" on page 396.

`IsConsoleStation`	Indicates whether the Station is a Console Station. Returns: • `0` = The Station is not a Console Station • `-1` = The Station is a Console Station
`LastSelectedPointID`	Returns the point ID of the last selected point.

Property	Description
SecurityLevel	Returns the user's security level. The values are: • 0 = View Only • 1 = Ack Only • 2 = Operator • 3 = Supervisor • 4 = Engineer • 5 = Manager
SecurityLevelAcronym	Returns the acronym (text representation) for the user's security level, for example: Oper.
SecurityType	Returns the Station security type for the current connection. The values can either be: • 0 = Station-based security • 1 = Operator-based security • 2 = Operator-based security (Windows users allowed) • 3 = Operator-based security (Windows users allowed, Signon Manager enabled) • 4 = Operator-based security (Windows users allowed, single signon)
ServerName	In the case of a Flex Station, returns name of the server to which the Station is connected. In the case of a Console Station, returns the name of the Console Station. (To learn about Console Stations, see the *Configuration Guide*.)
ServerVersion	Returns the version number of the server to which the Station is connected.
StationAcronym	Returns the number/identifier of the Station. For a Console Station the format is "CSTN02-1". For a Flex Station, this is of the form "STN01"
StationNumber	Returns the Station number.
Time	Returns the time that appears in the Status Line.

Methods

None.

Events

None.

Status Line boxes

Box (identifier)	Status values
Alarm Status (1)	• 0 = No Alarms. The background color should be transparent. • 1 = Acknowledged. There is at least one alarm, but they have all been acknowledged. The background color should be set to the color indicated in the corresponding Field Color property value. • 2 = UnAcknowledged Alarms. There is at least one unacknowledged alarm. The box should blink between a transparent background color and the specified Field Color property value.
System Status (2)	• 0 = Normal Comms. The background color should be transparent. • 1 = System Status Alarm Acknowledged. There is at least one system alarm, but they have all been acknowledged. The background color should be set to the color indicated in the corresponding Field Color property value. • 2 = UnAcknowledged System Status Alarms. There is at least one unacknowledged system alarm. The box should blink between a transparent background color and the specified Field Color property value.
Message Status (3)	• 0 = No Messages. There are no messages. The background color should be transparent. • 1 = Messages Acknowledged. There is at least one message, but they have all been acknowledged. The background color should be set to the color indicated in the corresponding Field Color property value. • 2 = UnAcknowledged Messages. There is at least one unacknowledged message. The box should blink between a transparent background color and the specified Field Color property value.

Box (identifier)	Status values
Alert Status (4)	• 0 = Normal Operation. Station is operating normally. The background color should be transparent. • 1 = Alert Acknowledged. There is at least one alert, but they have all been acknowledged. The background color should be set to the color indicated in the corresponding Field Color property value. • 2 = UnAcknowledged Alert. There is at least one unacknowledged alert. The box should blink between a transparent background color and the specified Field Color property value.
Cluster Server Status (5)	• -1 = Not A Console Station. The current connection is not to a console Station. Hence this box does not apply. • 0 = Connection OK. The current connection to the cluster server is operating normally. The background color should be transparent. • 1 = Connection lost. The connection to the cluster server has been lost. The background color should be set to the color indicated in the corresponding Field Color property value. • 2 = Connection synching. There is a valid, operating connection to the cluster server but the console station is currently synchronizing with the cluster server. The box should blink between a transparent background color and the specified Field Color property value.

Shape object

Properties

Property	Description
`autoReloadContent`	Returns or sets automatic reloading of the shape when there is a change in the value of a shape or any of its custom properties. The values are: • `True` = Automatic reload is enabled (default) • `False` = Automatic reloading is disabled Use the ReloadContent method to reload the shape if this property is set to `False`.
`customProperties.Count`	Returns the number of custom properties.
`customProperties.Item(`*n*`).type`	Returns the type of the nth custom property.
`customProperties.Item(`*n*`).name`	Returns the name of the nth custom property.
`customProperties.Item(`*n*`).value`	Returns the value of the nth custom property.

`Data*`
`Quality*`
`AlarmSeverity`
	These properties only exist if one or more point parameters are listed on the object's Script Data tab. See "Properties and methods of points on the Script Data tab" on page 306.

`id`	Returns the object's name.
`numberOfShapesAnimated`	Only applicable if the shape sequence is animated. Returns or sets the number of shapes included in an animated shape sequence.
`parentElement`	Returns the object's parent, which is either the group (if it is part of a group) or the page.
`rotation`	Returns or sets the object's rotation angle.
`shapefile.description`	Returns the description of the shape file.
`shapefile.hasScripts`	Returns `True` if the shape file has scripts. If not, it returns `False`.
`shapefile.numberOfShapes`	Returns the total number of shapes contained in the shape file.
`shapefile.title`	Returns the title of the shape file.
`shapefile.useFirstShapeForBadValue`	Returns `True` if the first shape in the shape sequence is used to indicate a bad value. If not, it returns `False`.

Property	Description
src	Returns or sets the name of the currently displayed shape file. Note that value can only be set if the shape is linked to the display. After setting this property, it is not possible to reference the subcontent of the shape object. If you want to do this, you must create a script in the onshapeload event of the shape to perform any operations that need to reference the subcontent of the shape object.
style	Returns or sets the object's style-related properties. See "style object" on page 407.
styleClass	Returns or sets the object's style. See "About style sheets" on page 108.
title	Returns or sets the object's title (ToolTip). To create a multi-line ToolTip, insert `Chr(10)` between each line, for example: `"Line1" & Chr(10) & "Line2"`
value	Returns or sets the number of the shape in the shape sequence. For example, a value of 2 indicates the second shape in the sequence. After setting this property, it is not possible to reference the subcontent of the shape object. If you want to do this, you must create a script in the onshapeload event of the shape to perform any operations that need to reference the subcontent of the shape object.

Methods

GetCustomProperty method

GetStyleProperty method

Objects method

ReloadContent method

SetCustomProperty method

Events

onactivate event

onblur event

onclick event

oncontextmenu event

ondatachange event

ondblclick event
ondeactivate event
onfocus event
onkeydown event
onkeypress event
onkeyup event
onmousedown event
onmouseenter event
onmouseleave event
onmousemove event
onmouseout event
onmouseover event
onmouseup event
onoperatorkey event (only available from within the shape file)
onshapeload event (only available from within the shape file)
onshapeunload event (only available from within the shape file)
onupdate event (only applicable to animated shape sequences)

Station object

Description

The Station object is the root object in the HMIWeb object model. From the Station object you can access information about the connection (the RuntimeStatus object), Station-wide settings (such as timers), and to access the Station windows (the StationWindows object).

Attention

Page-level scripts (scripts in the current display) access the Station object using `window.external.Parent`. See "Accessing Station-related properties in page-level scripts" on page 300.

Properties

Property	Description
`CommandZoneText`	Returns or sets the command that appears in the Command Zone. When used to set a command, the user must press **ENTER** to execute the command.
`ConnectionFile`	Returns the filename (and path) of the Station setup file (*.stn) used for the current connection to the server.
`Dictionary`	Returns the Dictionary object.
`MenuVisible`	Sets the visibility of Station's Menu. The values are either `True` (visible) or `False` (hidden).
`MessageZoneText`	Returns or sets the text that appears in the Station's Message Zone.
`RuntimeStatus`	Returns the RuntimeStatus object.
`ShowMessageBoxes`	Sets the visibility of internal Station message boxes. The values are either `True` (visible) or `False` (hidden). Set this property to `False` when Station is automated and there are no users to see message boxes or if message boxes would distract users. When this property is `False`, message boxes are answered with the default button. Scripts that run in this environment should check this property before showing a message box as only internal Station message boxes are affected by this object's setting.
`StationWindows`	Returns the StationWindows object.

Property	Description
TemporaryMessageZoneText	Sets the text of a temporary message that appears in the Station's Message Zone for 5 seconds (after which it is cleared). **Note:** If you set this property while there is a normal message in the Message Zone (represented by the MessageZoneText property), the temporary message will replace the normal one. Conversely, if a normal message is generated while the temporary message is in the Message Zone, the normal message will replace the temporary one.
Title	Returns the title that appears in the title bar of the Station window.
WindowMode	Returns Station's window mode, which can be: • 0 = Single window • 1 = Multi-window

Remarks

- This object is the best way to access Station wide methods and properties. It provides a way to write scripts that will work consistently in multi-window, single-window and browser-based environments.

Methods

CancelOperatorCommand method

Connect method

CreateTimer method

DisplayExists method

ExecuteOperatorCommand method

FindFile method

FlagUserInteraction method

GetConnectionProperty method

GetResponse method

InvokeCommand method

InvokePopup method

KillTimer method

LogMessage method

MessageBox method

Quit method

RequestServerLRN method

RequestTask method

SetCommandUIState method

SetTitleBarFormat method

SetUserVersionInfo method

Shell method

Events

OnAlarm event

OnAppStartup event

OnAppStatusUpdate event

OnBeforeConnect event

OnBeforeOKBKeyPressed event

OnBlink event

OnConnect event

OnCustomEvent event

OnDictionaryValueChanged event

OnDisconnect event

OnDisplayNotFound event

OnMenu event

OnOKBKeyPressed event

OnOperatorChange event

OnOperatorCommand event

OnPageLoad event

OnPeriodicUpdate event

OnQuit event

OnResponse event

OnSilenceIKB event

OnTimer event

StationDataRepository object

Description

The StationDataRepository object is a collection of named system custom properties within the scope of Station, which can be used by any script.

Properties

Property	Description
`Count`	Returns the number of properties in the repository.
`autoReloadContent`	Returns or sets automatic reloading of the custom property when there is a change in the value. The values are: • `True` = Automatic reload is enabled (default) • `False` = Automatic reloading is disabled

Remarks

- By default, system custom properties remain in the Station data repository for the life of Station. It is therefore important to use the Remove method to remove custom properties if they are no longer required.
- To improve the performance if you are changing the value of a lot of properties, you should set this back to `True`.

Methods

Add method

GetValue method

PutValue method

Remove method

Example

This example shows how to add the system custom property called “Custom2” and assign it as a point type and assign the default value as “sinewave”.

```
Window.External.StationDataRepository.Add(“Custom2”, “Point”,
   “sinewave”)
```

StationWindows object

Description

The StationWindows object is a collection of Application objects; one for each Station window. For example, if you have a multi-window Station with four windows, there are four instances of the Application object. Single-window Station contains only one instance of the Application object.

In a multi-window Station, it is possible for scripts on one display to refer to objects on another display. The StationWindows objects provides access to each Station window.

Properties

Property	Description
`ActiveWindow`	Returns a reference to the currently active window. This reference is equivalent to the Application object for that window.
`ActiveWindowIndex`	Returns the index number of the active window.
`Count`	Returns the number of windows in the StationWindows collection. This corresponds to the number of windows currently displayed.

Methods

Item method

Examples

To access the StationWindows collection from script on a display page, you must use the following name form:

```
window.external.Parent.StationWindows
```

For example, to find the name of the display in the current active window you use the following:

```
msgbox
   window.external.Parent.StationWindows.ActiveWindow.CurrentPag
   e
```

You can write a script to loop through the Station windows you have open, find the one you want and access the objects in that display. To do this, you use:

```
Dim stationwindow
for each stationwindow in window.external.parent.StationWindows
   MsgBox stationwide.Title
if(stationwindow.Title = “mypage.htm”) then
   MsgBox
   stationwindow.Application.Page.document.getElementById(“myobj
```

```
      ect").Value
      endif
   next
```

style object

Description

The style object is a standard DHTML object that represents many appearance-related properties of an object, such as its position, line thickness and font.

This example shows how to use the style object to change the x-coordinate (horizontal position) of an alphanumeric called “alpha3”:

```
alpha3.style.pixelLeft = 30
```

For a detailed description of the style object, go to Microsoft’s MSDN Web site at `http://msdn.microsoft.com/library/default.asp?url=/workshop/author/dhtml/reference/objects/obj_style.asp`

Properties

The following table only describes properties commonly used in display scripts.

Property	Description
`cursor`	Returns or sets the appearance of the mouse pointer when it moves over the object. Values include `crosshair`, `hand` and `help`.
`font`	Returns or sets the object’s font-related properties. For example: `style.font = "italic small-caps bold 12pt serif"`

`pixelHeight`
`pixelWidth`
`pixelLeft`
`pixelTop`
	Return or set the object’s position and height/width. See “Specifying dimensions and position” on page 321.

`visibility`	Returns or sets the object’s visibility setting: • `inherit` = The object inherits the visibility setting of its parent • `hidden` = The object is invisible • `visible` = The object is visible

Example scenarios

Creating a moving object

Text object

Properties

Property	Description

`Data*`
`Quality*`
`AlarmSeverity`
	These properties only exist if one or more point parameters are listed on the object's Script Data tab. See "Properties and methods of points on the Script Data tab" on page 306.

`disabled`	Returns or sets operator control of the object, the value is either `True` or `False`.
`fillColor`	Returns or sets the object's fill color. See "Setting and returning colors" on page 303.
`fillColorBlink`	Returns or sets the blinking state of the object's fill color. The value is either `True` or `False`.
`id`	Returns the object's name.
`lineColor`	Returns or sets the object's line color. See "Setting and returning colors" on page 303.
`lineColorBlink`	Returns or sets the blinking state of the object's line color. The value is either `True` or `False`.
`parentElement`	Returns the object's parent, which is either the group (if it is part of a group) or the page.
`rotation`	Returns or sets the angle (in degrees) about which the object is rotated.
`style`	Returns or sets the object's style-related properties. See "style object" on page 407.
`styleClass`	Returns or sets the object's style. See "About style sheets" on page 108.
`textColor`	Returns or sets the object's text color. The color can be either the RGB value (24-bit) or one of the predefined "Setting and returning colors" on page 303.
`textColorBlink`	Returns or sets the blinking state of the object's text color. The value is either `True` or `False`.
`title`	Returns or sets the object's title (ToolTip). To create a multi-line ToolTip, insert `Chr(10)` between each line, for example: `"Line1" & Chr(10) & "Line2"`
`value`	Returns or sets the text displayed in the object.

Methods

blur method

click method

focus method

GetStyleProperty method

Events

onactivate event

onblur event

onchange event

onclick event

oncontextmenu event

ondatachange event

ondblclick event

ondeactivate event

onfocus event

onkeydown event

onkeypress event

onkeyup event

onmousedown event

onmouseenter event

onmouseleave event

onmousemove event

onmouseout event

onmouseover event

onmouseup event

onupdate event

TitleToolbar object

Description

The TitleToolbar object is a child object of the Trend object.

For more information about the Trend object model, see “Overview of the trend object model” on page 336.

Properties

Property	Description
`visible`	Sets the visibility of the chart toolbar.

Methods

GetItemVisible method

SetItemVisible method

Events

None.

Trend object

Description

The trend object is the top level object of the trend control. It contains a number of objects that provide a greater level of control over the trend's features.

For more information about the Trend object model, see "Overview of the trend object model" on page 336.

Properties

Property	Description
`AutoRuntimeDataRequest`	Returns or sets the automatic data request when adding multiple points to a trend at runtime. True = Automatic data request at runtime is enabled. False = Automatic data request at runtime is disabled
`ChartArea`	Returns the ChartArea object.
`ChartToolbar`	Returns the ChartToolbar object.
`Legacy`	For backward compatibility. Returns the Legacy object which replaces the chart object.
`Legend`	Returns the Legend object.
`NumericHistory`	Returns the NumericHistory object.
`TimeSelectorPosition`	Returns or sets the current position of the time selector. The positions are: • `0` = SHOW_RIGHT • `1` = SHOW_LEFT • `2` = SHOW_CENTER • `3` = SHOW_LEFT_RIGHT • `4` = HIDDEN
`TitleText`	Sets the title text in the trend's title bar.
`TitleToolbar`	Returns the TitleToolbar object.
`View`	Returns or sets the current trend view. The views are: • `0` = Trend with full legend • `1` = Trend with tabular history • `2` = Trend with events • `3` = Trend with mini legend

Methods

GetSampleInterval method

RuntimeDataRequest method

SetPeriod method

SetSampleInterval method

Events

OnEventDataTipDisplayed event

OnPlotDetailsChanged event

OnTimSelectorPositionChanged event

OnPause event

OnPlotDataBufferFull event

OnReferenceCursorSet

OnResume

OnRequestDataTip event

OnRequestEventInfo event

OnRequestDataTip event

OnScrollXAxis

OnScrollYAxis event

OnTrendLoadComplete event

OnUnZoomed event

OnZoom event

Vector Graphic object

Description

A vector graphic object is a “line-based” object such as a rectangles or curve. See “About vector graphic objects” on page 222.

Properties

Property	Description

`Data*`
`Quality*`
`AlarmSeverity`
	These properties only exist if one or more point parameters are listed on the object’s Script Data tab. See “Properties and methods of points on the Script Data tab” on page 306.

`fillColor`	Returns or sets the object’s fill color. See “Setting and returning colors” on page 303.
`fillColorBlink`	Returns or sets the blinking state of the object’s fill color. The value is either `True` or `False`.
`FillStyle`	Returns or sets the object’s fill style. The value can be: • `0` = Solid • `1` = Transparent • `2` = Gradient
`GradientFillColor`	Returns or sets the color used for the gradient fill. The color can be either the RGB value (24-bit) or one of the predefined “Setting and returning colors” on page 303.
`GradientFillFocus`	Returns or sets the *focus* of the object’s gradient fill. (The focus defines the relative position between the object’s edges.) The value is between `0` and `100`. For example, `50` represents the midpoint of the object.
`GradientFillStyle`	Returns or sets the object’s gradient fill style. The value can be: • `0` = Vertical • `1` = Horizontal
`id`	Returns the object’s name.
`Level`	Returns or sets the level fill value. A floating point value between `0.0` and `1.0`.
`levelFillColor`	Returns or sets the object’s level fill color. The color can be either the RGB value (24-bit) or one of the predefined “Setting and returning colors” on page 303. Note that the color is only visible if the LevelFilled property is `True`.

Property	Description
`LevelFilled`	Returns a value indicating whether the object has a level fill color. (defined by the levelFillColor). The value is either `True` or `False`.
`lineColor`	Returns or sets the object's line color. See "Setting and returning colors" on page 303.
`lineColorBlink`	Returns or sets the blinking state of the object's line color. The value is either `True` or `False`.
`LineStyle`	Returns or sets the object's line style. The value can be: • `0` = Transparent • `1` = Solid • `2` = ShortDash • `3` = Dash • `4` = LongDash • `5` = Dot • `6` = DashDot • `7` = DashDotDot
`LineWidth`	Returns or sets the object's line width.
`Nodes`	Returns the Nodes object, the collection that represents the object's nodes.
`parentElement`	Returns the object's parent, which is either the group (if it is part of a group) or the page.
`Rotation`	Returns or sets the angle (in degrees) about which the object is rotated.
`StartArrow` `EndArrow`	Return or set the object's line-ending styles. Only applicable to "open" objects, such as lines and curves. The value can be: • `0` = No arrow • `1` = Open arrow • `2` = Closed arrow • `3` = Oval
`style`	Returns or sets the following style-related properties: • Position • Cursor • Visibility See "style object" on page 407. Other properties typically set by the style object are specified using object-specific properties.

Property	Description
`styleClass`	Returns or sets the object's style. See "About style sheets" on page 108.
`title`	Returns or sets the object's title (ToolTip). To create a multi-line ToolTip, insert `Chr(10)` between each line, for example: `"Line1" & Chr(10) & "Line2"`

Methods

GetStyleProperty method

Events

onactivate event

onblur event

onchange event

onclick event

oncontextmenu event

ondatachange event

ondblclick event

ondeactivate event

onfocus event

onkeydown event

onkeypress event

onkeyup event

onmousedown event

onmouseenter event

onmouseleave event

onmousemove event

onmouseout event

onmouseover event

onmouseup event

onupdate event

window object

Description

Represents an open window.

Properties

Property	Description
external	See “Application object” on page 342
event	See “event object” on page 362

Methods

setTimeout method

setInterval method

clearInterval method

clearTimeout method

Examples

XAxis object

Description

The Xaxis object is a child object of the ChartArea object. It contains properties for controlling and manipulating the x-axis of the char area.

For more information about the Trend object model, see “Overview of the trend object model” on page 336.

Properties

Property	Description
`ApplyScrollLimits`	Returns or sets the application of scrolling limits.
`BackgrounColor`	Returns or sets the background color of the axis.
`BorderColor`	Returns or sets the color of the axis border.
`BorderLayout`	Returns or sets the placement of borders around the axis perimeter: • `0` = No border • `1` = Top border • `2` = Bottom border • `4` = Left border • `8` = Right border • `15` = Border around entire perimeter
`BorderStyle`	Returns or sets the border style of the axis: • `0` = No border • `1` = Flat border (the default) • `2` = Sunken border • `3` = Raised border • `4` = 3D border
`BorderVisible`	Returns or sets the visibility of the axis border.
`BorderWidth`	Returns or sets the border width in pixels.
`LabelColor`	Returns or sets the axis label color.
`LabelFont`	Returns or sets the axis label font.
`LabelVisible`	Returns or sets the visibility of the axis label.
`MarkerFont`	Returns or sets the axis marker text.
`MarkerLabelsVisible`	Returns or sets the visibility of the axis marker label.
`MarkerLabelsWrapped`	Returns or sets the placement of the date/time. If set to `True`, the date and time is split over two lines. If set to `False`, then the date and time is displayed on one line.

Property	Description
MarkerLabelColor	Returns or sets the color of the axis marker text.
ScrollBarVisbile	Returns or sets the visibility of the axis scroll bar.
ScrollLineAmount	Returns or sets the line amount as a percentage of the visible range. A scroll line occurs when the scroll but is clicked. The default value is 0.1 (10%).
ScrollPageAmount	Returns or sets the scroll page amount as a percentage of the visible range. A scroll line occurs when the area between the thumbtrack and the scroll button is clicked. The default value is 0.5 (50%).
ScrollPosition	Returns the current scroll bar position.
Visible	Returns or sets the visibility of the x-axis.

Methods

GetInPlaceEditing method

GetRange method

Scroll method

SetInPlaceEditing method

Events

None.

YAxis object

Description

The Yaxis object is a child object of the ChartArea object. It contains properties for controlling and manipulating the y-axis of the char area.

For more information about the Trend object model, see “Overview of the trend object model” on page 336.

Properties

Property	Description
`ApplyScrollLimits`	Returns or sets the application of scrolling limits.
`BackgrounColor`	Returns or sets the background color of the axis.
`BorderColor`	Returns or sets the color of the axis border.
`BorderLayout`	Returns or sets the placement of borders around the axis perimeter: • `0` = No border • `1` = Top border • `2` = Bottom border • `4` = Left border • `8` = Right border • `15` = Border around entire perimeter
`BorderStyle`	Returns or sets the border style of the axis: • `0` = No border • `1` = Flat border (the default) • `2` = Sunken border • `3` = Raised border • `4` = 3D border
`BorderVisible`	Returns or sets the visibility of the axis border.
`BorderWidth`	Returns or sets the border width in pixels.
`LabelColor`	Returns or sets the axis label color.
`LabelFont`	Returns or sets the axis label font.
`LabelVisible`	Returns or sets the visibility of the axis label.
`MarkerFont`	Returns or sets the axis marker text.
`MarkerLabelsVisible`	Returns or sets the visibility of the axis marker label.
`MarkerLabelsWrapped`	Returns or sets the placement of the date/time. If set to `True`, the date and time is split over two lines. If set to `False`, then the date and time is displayed on one line.

Property	Description
MarkerLableColor	Returns or sets the color of the axis marker text.
ScrollBarVisbile	Returns or sets the visibility of the axis scroll bar.
ScrollLineAmount	Returns or sets the line amount as a percentage of the visible range. A scroll line occurs when the scroll but is clicked. The default value is 0.1 (10%).
ScrollPageAmount	Returns or sets the scroll page amount as a percentage of the visible range. A scroll line occurs when the area between the thumbtrack and the scroll button is clicked. The default value is 0.5 (50%).
ScrollPosition	Returns the current scroll bar position.
Visible	Returns or sets the visibility of the x-axis.

Methods

GetInPlaceEditing method

GetRange method

Scroll method

SetInPlaceEditing method

Events

None.

Methods

The methods are listed in alphabetical order.

Attention

Because of the sheer size and complexity of the DOM, this help only describes methods that are either specific to HMIWeb displays, or are particularly important.

Add method

Applicable to

Dictionary object

StationDataRepository object

DisplayDataRepository object

Description

Adds a new item to the object.

Syntax

```
Add name,value
```

Part	Description
name	The name of the item. The name is case-sensitive.
value	The value of the item. The value can be of any form, such as text or numeric.

Return values

None.

Example

This example adds an item “Voltage” to the Dictionary and sets its value to “12.3”.

```
window.external.Dictionary.Add "Voltage", 12.3
```

This example adds a system custom property “Custom1” to the Station Data Repository.

```
window.external.stationDataRepository.Add “Custom1” “Point”
```

This example adds a system custom property “Custom1” to the display Data Repository.

```
displayDataRepository.Add “Custom1” “Point”
```

AddNewPlot method

Applicable to

Legacy object

Description

Adds a new plot to a chart, but does not add data to it.

Syntax

```
AddNewPlot label, ylow, yhigh, ylabel, xlow, xhigh, xlabel
```

Part	Description
label	The label for the plot's ToolTip. See "Plot ToolTips" on page 224.
ylow *yhigh*	The y-axis low and high limits.
ylabel	The y-axis label.
xlow *xhigh*	The x-axis low and high limits.
xlabel	The x-axis label.

Return values

The plot's ID, which is assigned by the Chart object.

Remarks

- Use the AddPlotData method to add data to the plot.
- The method will fail if 32 plots have already been added to the chart, or if not enough memory can be allocated.

Example

This example shows how to a add a new plot to chart003, and then add data to it. (Note that "g_PlotID1" is a global page variable that used in later scripts to identify the plot.)

This adds the plot using the AddNewPlot method.

```
g_PlotID1 = chart003.Legacy.AddNewPlot("Tank1 level", 0, 100,
   "Meters", XRangeMin, XRangeMax, "Time")
```

Later scripts add data to the plot using the AddPlotData method.

```
dim XData(60), YData(60)
```

```
For n = 0 to 60
  XData(n) = DateAdd("s", n, #20/5/2002 12:00:00 PM#)
  YData(n) = (20*sin(n)) + 50
Next
chart003.Legacy.AddPlotData g_PlotID1, YData, XData
```

Later scripts can also, for example, clear the plot.

```
chart003.Legacy.ClearPlotData g_PlotID1
```

AddPlotData method

Applicable to

Legacy object

Description

Adds data to an existing plot.

Syntax

```
AddPlotData plotID, plotydata, plotxdata
```

Part	Description
plotID	The plot's ID.
plotydata *plotxdata*	The x-and y-axis values for each sample point of the plot. See "Specifying a plot's data" on page 426.

Return values

None.

Remarks

- Currently, there is no simple mechanism for specifying a point parameter, such as fast history, and the number of samples—as you do when you add the chart to the display. If you need to obtain point parameter samples, you could write a network application that creates the required data array. For details, see the *Application Development Guide*.

Example

This example shows how to add a new plot to chart003, and then add data to it. (Note that "g_PlotID1" is a global page variable that used in later scripts to identify the plot.)

This adds the plot using the AddNewPlot method.

```
g_PlotID1 = chart003.Legacy.AddNewPlot("Tank1 level", 0, 100,
   "Meters", XRangeMin, XRangeMax, "Time")
```

Later scripts add data to the plot using the AddPlotData method.

```
dim XData(60), YData(60)
For n = 0 to 60
  XData(n) = DateAdd("s", n, #20/5/2002 12:00:00 PM#)
  YData(n) = (20*sin(n)) + 50
Next
chart003.Legacy.AddPlotData g_PlotID1, YData, XData
```

Specifying a plot's data

You can specify x-axis data (*plotxdata*) by either:

- Specifying a value for each x value as an element in an array, that is: [*x1, x2, x3, …*]
- Specifying the first x value and an x increment, that is: [*xstart, xincrement*] . (Note that *xstart* and *xincrement* must be specified using the same units. For example, if *xstart* is a date then *xincrement* must also be a date.)

AddString method

Applicable to

Combobox object

Description

Adds an item to the combobox's list.

Syntax

```
AddString item
```

Part	Description
item	The item being added.

Return values

None.

Remarks

- To use this method, you must set the combobox's **Type of database link** property to `None`. (On the Data tab of the Properties Window.)
- Each item is assigned an index number, the index of the first item being 0. The item number is used with the DeleteString method to remove an item.
- By default, the first item in the list is selected when the combobox is initially displayed.

Example

This example adds three items to the combobox "SetSpeed".

```
SetSpeed.AddString "Slow"
SetSpeed.AddString "Medium"
SetSpeed.AddString "Fast"
```

Autoscale method

Applicable to

Plot object

Plots object

Description

Applies automatic scaling to the plot. If the trend is not using a single scale for all plots, then the plots are scaled to fill the Y-axis.

Syntax

```
Autoscale
```

Remarks

- This only applies to the data currently visible in the trend. If the user scrolls or the trend is updated with live updates, you may want to autoscale again.

Blue method

Applicable to

Application object

Description

The Blue method is a utility function that can be used in any script. It returns the blue component of a color value, expressed as a number between 0 (no blue) and 255 (100% blue). Normally used in conjunction with the Red and Green methods.

Syntax

```
Blue RGBvalue
```

Part	Description
RGBvalue	The color value from which the blue component is returned.

Return values

The blue component of the color.

Example

This example obtains the blue component of the fill color of “Rect3”.

```
BlueValue = window.external.Blue(Rect3.FillColor)
```

blur method

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Hyperlink object

Indicator object

Picture object

Pushbutton object

Shape object

Text object

Vector Graphic object

Description

Standard DHTML method that causes the object to lose focus and fire an onblur event for the object.

Syntax

```
blur
```

Return values

None.

CancelChange method

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Pushbutton object

Description

Cancels a change made by the user to the object.

Syntax

```
CancelChange
```

Return values

None.

Remarks

- The CancelChange method can only be used within an onchange event. Together, they enable the change to be validated before passing it on to the server.

Example

This example compares the user-entered value with the current limit and cancels the change if the value is greater than the limit.

```
Sub alpha1_onchange
    `calculate current limit.
    .
    .
    .
  If alpha1.value > Limit Then
    window.external.MessageZoneText = "Value must be below " &
   CStr(limit) & "."
    alpha1.CancelChange
  End If
End Sub
```

CancelEvent method

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Pushbutton object

Description

Cancels the last event fired by the user.

Syntax

```
CancelEvent
```

Return values

None.

Remarks

- The event is typically used to cancel the user's click on a pushbutton because it enables the click to be validated before being passed to the server.

Example

This example, attached to "Pushbutton1", checks the user's security level and cancels the click if the level is below Engineer.

```
Sub PushButton1_onclick()
  If window.external.SecurityLevel < 4 Then
    window.external.MessageZoneText = "Only an Engineer or
   Manager can do this"
    pushbutton1.CancelEvent
  End If
End Sub
```

clearInterval method

Applicable to

window object

Description

Cancels the interval previously started using the SetInterval method. See “setInterval method” on page 532.

Syntax

```
window.clearInterval(iIntervalID)
```

Part	Description
iIntervalID	A required integer that specifies the interval to cancel. This value must have been previously returned by the setInterval method.

Return values

None.

Remarks

- For more information about timers see “Timers” on page 291.

Example

- See “Creating an animation with a shape sequence” on page 650

CancelOperatorCommand method

Applicable to

Application object

Station object

Description

Cancels the command entered by the user in the Station's Command Zone.

Syntax

```
CancelOperatorCommand
```

Return values

None.

Remarks

- The CancelOperatorCommand method can only be used within an OnOperatorCommand event. Together, they enable the command to be validated before passing it on to the server.
- This method is not available in eServer premium.

Example

```
Sub OnOperatorCommand(bstrCommand)
  If bstrCommand = "bye" Then
    window.external.Parent.MessageZoneText = "Log out not
   permitted"
    CancelOperatorCommand
  End If
End Sub
```

CancelResponse method

Applicable to

Application object

Description

Cancels the current GetResponse method (all variants), clears the Message Zone and ignores any user input in the Command Zone.

Syntax

```
CancelResponse
```

Return values

None.

Remarks

- This method is not available in eServer premium.

ClearPlotData method

Applicable to

Legacy object

Description

Clears a plot's data from a chart.

Syntax

```
ClearPlotData plotID
```

Part	Description
plotID	The plot's ID.

Return values

None.

Example

This example shows how to add a new plot to chart003, and then clear it. (Note that "g_PlotID1" is a global page variable that used in later scripts to identify the plot.)

This adds the plot using the AddNewPlot method.

```
g_PlotID1 = chart003.Legacy.AddNewPlot("Tank1 level", 0, 100,
   "Meters", XRangeMin, XRangeMax, "Time")
```

Later scripts can clear the plot.

```
chart003.Legacy.ClearPlotData g_PlotID1
```

Remarks

- From the user's point of view, the plot is removed from the chart. However, the plot can be repopulated (restored) with the AddPlotData method, providing the plot's ID is still valid.
- Use the RemovePlot method to delete the plot.

clearTimeout method

Applicable to

window object

Description

Cancels a time-out that was set with the setTimeout method. See “setTimeout method” on page 549.

Syntax

```
window.clearTimeout(iTimeoutID)
```

Part	Description
iTimeoutID	A required integer that specifies the time-out setting returned by a previous call to the setTimeout method.

Return values

None.

Remarks

- For more information on timers see “Timers” on page 291.

click method

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Hyperlink object

Indicator object

Page object

Picture object

Pushbutton object

Shape object

Text object

Vector Graphic object

Description

Standard DHTML method that simulates an onclick event for the object.

Syntax

```
click
```

Return values

None.

Example

```
Sub alpha1.onchange
  If alpha1.value = "yes" Then
    pushbutton1.click
  End If
End Sub
```

Connect method

Applicable to

Application object

Station object

Description

Specifies the Station setup file (`.stn`) used by Station when it is started. This method is required by an automated Station because Station does not automatically use `default.stn`, even if it is in the same folder as the Station executable.

Syntax

```
Connect filename
```

Part	Description
filename	The name of the Station setup file, including the path. If you specify a null string (“”), the method uses `default.stn`.

Return values

None.

Remarks

- You can use the OnBeforeConnect event and the GetConnectionProperty method to check whether the connection properties in the setup file are appropriate before using this method.
- This method is not available in eServer premium.

Example

This example shows how to start Station using a setup file called “special.stn”.

```
Window.External.Parent.Connect "C:\Honeywell\Client\Station\
   special.stn"
```

This example shows how to start Station using the default setup file.

```
Window.External.Parent.Connect ""
```

Copy method

Applicable to

Legacy object

Description

Copies currently visible data to the clipboard. All data associated with the plot(s), including current values, dates/times, and point names/parameters, is copied. The copied data is in standard text format, which is suitable for pasting into applications such as Microsoft Excel.

Syntax

```
Copy
```

Return values

None.

Remarks

- If the chart has scrollbars, operators can determine what data is visible—and therefore what the method copies to the clipboard.

Example

After setting up a chart (called “chart001”) and then adding data to it, you would use the following code to copy the data to the clipboard.

```
chart001.Legacy.Copy
```

CopyDataToClipboard method

Applicable to

ChartArea object

Description

Copies the data for all visible plots to the clipboard.

Syntax

```
CopyDataToClipboard
```

Return values

None.

Remarks

- This method copies all of the plot samples to the Windows clipboard. From the clipboard, the data can be pasted into a text document or Microsoft Excel spreadsheet.

Example

After setting up a trend (called “trend001”) and then adding data to it, you would use the following code to copy the data to the clipboard.

```
trend001.CopyDataToClipboard
```

CreatePopupWindow method

Applicable to

Popup object

Description

Calls up a popup window or faceplate.

Syntax

```
CreatePopupWindow DisplayPath, Xpos, Ypos[, PositionType]
```

Part	Description
DisplayPath	The filename of the popup display or faceplate. If the file is not in one of Station's display folders, you must also include the full path, for example: `C:\private\mypopup.htm`.
Xpos *Ypos*	The x- and y-coordinates of the top-left of the popup relative to the parent display. (The coordinates of the top-left of the display are `1 1`.) If you specify `0, 0`, Station uses the default the positioning rules.
PositionType	Specifies the type of positioning used (if *Xpos* and *Ypos* are set to `0`): • `0` = Positioning rules for popups • `1` = Positioning rules for faceplates

Return values

None.

Example

This example calls up a popup called "Status" (which is in one of Station's display folders) and places it at the top-left of the display.

```
HDXPopupBehavior.CreatePopupWindow "Status.htm", 1, 1, 0
```

This example calls up the faceplate the analog point called "Tank1" and places it in the default position. (Where "sysdtlana_fp.htm" is the standard faceplate for an analog point.)

```
HDXPopupBehavior.CreatePopupWindow "sysdtlana_
   fp.htm?currentpoint=tank1", 0, 0, 1
```

CreateTimer method

Applicable to

Station object

Description

Attention

You should only use this CreateTimer method within Application Scripts.

Creates a timer object that will cause the Station, Application, and Page objects OnTimer events to be fired at a specified interval. To stop the timers, call the KillTimer method with the same id used to create the timer object. Creating a timer object with the same id as an existing timer object will kill the existing timer and create a new one.

Syntax

```
CreateTimer timerID, timerInterval
```

Part	Description
timerID	A unique number used to identify the timer. Multiple timers can exist. The OnTimer script can be coded to check the timerID and choose how to respond to individual timers.
timerInterval	The interval, in milliseconds, that the timer fires OnTimer events.

Remarks

- A maximum of 100 timers can exist at any one time.
- The timer's interval can be changed by calling the CreateTimer method a second time, specifying the original id, but a different interval.
- Timers with intervals of less than 100 ms are not recommended for performance reasons.

Example

This example fires the OnTimer event for the Station, Application, and Page objects every 5 seconds, and creates a timer with an id of 11.

```
window.external.Parent.CreateTimer 11, 500
```

DecreaseZoomLevel

Applicable to

PlotCanvas object

Description

Zooms out by approximately 20%.

Syntax

```
DecreaseZoomLevel
```

Returns values

None.

DeleteString method

Applicable to

Combobox object

Description

Deletes an item from the Combobox's list.

Syntax

```
DeleteString index
```

or

```
numItems = DeleteString (index)
```

Part	Description
index	The item's index number, which represents its position in the list. The index of the first item is `0`.
numItems	Returns the remaining number of items in the Combobox's list.

Return values

None.

Remarks

- By default, the first item in the list is selected when the combobox is initially displayed.

Example

This example deletes the item with index 1 from a combobox called "SetSpeed".

```
numItems = SetSpeed.DeleteString(1)
```

DisplayExists method

Applicable to

Application object

Station object

Description

Checks whether the specified display exists in Station's display folders (listed in the **Connection Properties** dialog box.)

Syntax

```
DisplayExists displayname
```

Part	Description
displayname	The name of the display. (The `.htm` or `.dsp` extension is not required.)

Return values

`True` if the display exists or `False` if it doesn't.

DPToLPX and DPToLPY methods

Applicable to

AppWindow object

Description

These two methods convert the X- and Y-coordinates, respectively, from resolution-dependent display coordinates (expressed in pixels) to HMIWeb Display Builder's logical coordinates. See "Specifying dimensions and position" on page 321.

Syntax

```
DPToLPX(displaycoordx)
DPToLPY(displaycoordy)
```

Part	Description
displaycoordx *displaycoordy*	The display coordinate that is converted.

Remarks

- The LPToDPX and LPToDPY methods convert from logical coordinates to display coordinates.

Example

This example sets the logical coordinates of "Rect1".

```
Rect1.Left = window.external.AppWindow.DPToLPX(500)
Rect1.Top = window.external.AppWindow.DPToLPY(300)
```

ExecuteOperatorCommand method

Applicable to

Application object

Station object

Description

Executes an operator command as if it had been entered into Station's Command Zone by the user.

Syntax

```
ExecuteOperatorCommand command
```

Part	Description
command	The command that is executed.

Return values

None.

Remarks

- If you use ExecuteOperatorCommand within an OnOperatorCommand event, the command specified in ExecuteOperatorCommand will be executed and the user's command will be canceled.
- See the *Operators Guide* for the list of operator commands.

Example

This example calls up a display, `MainPlant.htm`.

```
window.external.Parent.ExecuteOperatorCommand "Pag MainPlant"
```

FindFile method

Applicable to

Application object

Station object

Description

Searches through the display paths for the specified file.

Syntax

```
FindFile file
```

Part	Description
file	The name of the file, including its extension.

Return values

The file's path (location), or `FALSE` if the file is not found (and the path is set to a null string).

Example

This example returns the location of Station's default toolbar definition file, "default.stb".

```
str = Window.External.Parent.FindFile "default.stb"
```

FindPopup method

Applicable to

Popup object

Description

Finds the open popup/faceplate whose URL contains the specified string.

Syntax

```
FindPopup string
```

Part	Description
string	The string that is searched for in all open popups. If the string appears in the URL of more than one popup, the method finds the oldest popup. In the case of a faceplate, you can specify the point ID because it is included the URL. For example, the URL of the faceplate for an analog point called “sinewave” would be: `…\sysDtlAna_fp.htm?CurrentPoint=sinewave` The method finds the oldest popup if you do not specify anything (a “null string”).

Return values

The popup window with the URL that matches the search criteria, or “nothing” if no popup matched the search criteria.

Example

This example closes the faceplate displaying information about a point called “sinewave”.

```
Dim objPopupWindow
set objPopupWindow = HDXPopupBehavior.FindPopup("sinewave")
if objPopupWindow is nothing then
  window.external.MessageZoneText = " Popup Window Not Found "
  Exit Sub
End if
objPopupWindow.close
```

focus method

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Hyperlink object

Indicator object

Picture object

Pushbutton object

Shape object

Text object

Vector Graphic object

Description

Standard DHTML method that causes the object to gain focus and fire an onfocus event for the object.

Syntax

```
focus
```

Return values

None.

Remarks

- We recommend that you don't use this method as it can interfere with the standard HMIWeb operation. We suggest that you use an click method instead. See "click method" on page 438.

Example

```
Sub page_onpagecomplete
  'set focus to first alpha on page
  alpha1.focus
End Sub
```

FlagUserInteraction method

Applicable to

Application object

Station object

Description

Resets Station's user idle timeout.

Syntax

```
FlagUserInteraction
```

Return values

None.

GetConnectionProperty method

Applicable to

Application object

Station object

Description

Returns the value of the specified Station connection property.

The connection properties are defined in the Station setup file (`*.stn`) used by Station when it is started.

Syntax

```
GetConnectionProperty property
```

Part	Description
property	The connection property.

Return values

The value of the connection property, or `VT_EMPTY` if the property does not exist.

Remarks

- This method is typically used in an OnBeforeConnect event to check whether the connection properties in a Station setup file (`*.stn`) are appropriate before using the file is used by the Connect method to start Station.
- This method is not available in eServer premium.

Example

This example shows how to determine a rotary Station's startup display.

```
Dim StartDisplay
StartDisplay =
   Window.External.Parent.GetConnectionProperty("LAN\
   StartupDisplay")
```

GetCustomProperty method

Applicable to

Shape object

Description

Returns the value of the specified custom property.

Syntax

```
GetCustomProperty type, name
```

Part	Description
type	The custom property's type.
name	The custom property's name.

Return values

The value of the custom property.

Remarks

- To learn about custom properties, see "About custom properties" on page 57.
- Use the SetCustomProperty method to set the value of a custom property.

Example

This example changes the fill color of the object called "rect1", which is part of "group1" in the shape file. The custom property is called "clickcolor" and its type is "color". Note that clickcolor must be either a named HTML color such as "RED" or an RGB value of the form "#*nnnnnn*".

```
Sub rect1_onclick
  me.fillColor = group1.parentElement.GetCustomProperty("color",
   "clickcolor")
End Sub
```

GetDataTipArrowColors method

Applicable to

EventBar object

Description

Returns the color of the arrows associated with multiple events in an event data tip.

Syntax

```
GetDataTipArrowColors (pclrArrow, pclrButtonFace)
```

Part	Description
pclrArrow	The arrow used in data tips.
pclrButtonFace	The background on which the arrow appears.

Return values

The color of the arrows and the background color in RGB format.

GetDataTipFormat method

Applicable to

ReferenceCursor object

Description

Returns the format of the data tip currently used for the specified chart type.

Syntax

```
GetDataTipFormat (ctType, strFormat)
```

Part	Description
ctType	The chart type.
strFormat	Specifies what information is shown when the mouse hovers above the reference cursor.

Return values

The data tip format which can be:

%x = X-axis value

%X = X-axis units

%y = Y-axis value

%Y = Y-axis units

Remarks

The default formats are:

Single bar chart = %x %X

Single scale line chart = %x %X

Line chart = %y %Y %x

GetDataTipVisible method

Applicable to

EventBar object

Description

Returns the visibility of data tips for an event.

Syntax

```
GetDataTipVisibility nEventID
```

Part	Description
nEventID	The ID of the event.

Return values

The visibility setting of the data tip, which can be:

`True` = Visible

`False` = Hidden

GetGridLinePen method

Applicable to

PlotCanvas object

Description

Returns the properties of the pen used to draw the grid lines.

Syntax

```
GetGridLinePen (nLineStyle, nLineWidth, clrLine)
```

Part	Description
nLineStyle	The line style of the grid lines in the plot.
nLineWidth	The width of the grid line (in pixels).
clrLine	The color of the plot.

Return values

The line style of the grid lines, which can be:

0 = Solid line

1 = Dashed line

2 = Dotted line

3 = Dashed/dotted line

4 = Dashed/double-dotted line

GetInPlaceEditing method

Applicable to

XAxis object

YAxis object

Description

Returns the “inplace editing” state for the axes. On the x-axis editing can be enabled or disabled for the date/time control, including the positioning of the control. On the y-axis, editing can be enabled or disabled for scaling.

Syntax

```
GetInPlaceEditing (pEnabled, pPosition)
```

Part	Description
pEnabled	Returns whether editing is enabled.
pPosition	Returns the position of the date/time control

Returns values

The editing state for the axes, which can be:

0 = Editing disabled

1 = Editing enabled

The position state, which can be:

0 = SHOW_RIGHT

0 = SHOW_TOP

1 = SHOW_LEFT

1 = SHOW_BOTTOM

2 = SHOW_CENTRE

3 = SHOW_LEFT_RIGHT

3 = SHOW_TOP_BOTTOM

4 = HIDDEN

GetItemVisible method

Applicable to

ChartToolbar object

TitleToolbar object

Description

Returns the visiblity of items on the toolbar.

Syntax

```
GetItemVisible (tbItem, vbItemVisible)
```

Return values

The toolbar items, which can be:

1 = The View trend drop-down button and the Show legend button.

2 = The View trend drop-down button

3 = The Show legend button

20 = The y-axis label, which indicates the units for the currently selected plot.

21 = This includes:

- Plot color indicator
- Plot selection combobox
- Scale configuration dropdown button
- Chart type dropdown button

22 = This includes the time period combo and the interval combo.

23 = The scale configuration dropdown button.

24 = The chart type selection dropdown button.

The visibility of the toolbar items which can be:

0 = Hidden

1 = Visible

GetLabels method

Applicable to

Plot object

Description

Returns the plot name, x-axis label, and y-axis label.

Syntax

```
GetLabels (strPlotLabel, strXLabel, strYlabel)
```

Part	Description
strPlotLabel	The plot label.
strXLabel	The x-axis label.
strYLabel	The y-axis label.

Returns values

The labels for the plot and x- and y-axis in string format.

GetLineStyle method

Applicable to

ReferenceCursor object

Description

Returns the line style for the reference cursor on the chart.

Syntax

```
GetLineStyle (clrLine, nWidth)
```

Part	Description
clrLine	The color of the reference line.
nwidth	The width of reference line.

Return values

The color and width of the reference line.

Remarks

The line style is solid for all lines more than 1 pixel thick.

GetPen method

Applicable to

Plot object

Description

Returns the pen style of the plot line.

Syntax

```
GetPen (nLineStyle, nLineWidth, clrLine)
```

Part	Description
nLineStyle	The line style of the plot:
nLineWidth	The width of the plot line (in pixels).
clrLine	The color of the plot.

Return values

The line style of the plot pen which can be:

0 = Solid line

1 = Dashed line

2 = Dotted line

3 = Dashed/dotted line

4 = Dashed/double-dotted line

GetPeriod method

Applicable to

Trend object

Description

Returns the period of the trend.

Syntax

```
GetPeriod (nMonths, nWeeks, nDays, pHours, pMinutes, pSeconds)
```

Part	Description
nMonths	The number of months in the period.
nWeeks	The number of weeks in the period.
nDays	The number of days in the period.
pHours	The number of hours in the period.
pMinutes	The number of minutes in the period.
pSeconds	The number of seconds in the period.

Return values

The period of the trend as an integer.

GetPlotColor method

Applicable to

Legacy object

Description

Returns the color of a plot.

Syntax

```
GetPlotColor plotID
```

Part	Description
plotID	The plot's ID.

Return values

The color of the plot specified by the plot ID.

Example

This example displays a message box, indicating the plot's alarm state which corresponds with the plot's color.

```
dim clrPlotID1
clrPlotID1 = chart003.Legacy.GetPlotColor(g_PlotID1)
window.external.MessageZoneText = "clrPlotID1 = " & clrPlotID1
If clrPlotID1 = vbRed Then
  textbox001.value = "In Alarm"
Else
  textbox001.value = "Out of Alarm"
End If
```

Remarks

- The color can be either the RGB value (24-bit) or a VBScript color constant (see "Setting and returning colors" on page 303).

GetPlotDiscrete method

Applicable to

Legacy object

Description

Returns the “line style” of a plot. (That is, whether or not lines appear between samples.)

Syntax

```
GetPlotDiscrete plotID
```

Part	Description
plotID	The plot’s ID.

Return values

The line style:

`1` = Lines appear between samples

`0` = No lines between samples

Example

This example checks whether lines appear between samples of the current plot, and adds the lines if they aren’t currently shown.

```
Dim nPlotID, bDiscrete
nPlotID = chart001.Legacy.CurrentPlot
bDiscrete = chart001.Legacy.GetPlotDiscrete nPlotID
if bDiscrete = 0 Then
  Chart001.Legacy.SetPlotDiscrete nPlotID, 1
End if
```

GetPlotMarkerStyle method

Applicable to

Legacy object

Description

Returns the “marker style” of a plot. (The symbol used to mark samples in the chart.)

Syntax

```
GetPlotMarkerStyle plotID
```

Part	Description
plotID	The plot’s ID.

Return values

The marker style, which can be:

`0` = None

`1` = X

`2` = Plus

`3` = Square

`4` = Diamond

`5` = Triangle

`6` = DownTriangle

`7` = Circle

`19` = SquareSolid

`20` = DiamondSolid

`21` = TriangleSoild

`22` = DownTriangleSoild

`23` = CircleSolid

Example

This example gets the marker style of the current plot.

```
Dim nPlotID, nMarkerStyle
nPlotID = chart001.Legacy.CurrentPlot
nMarkerStyle = chart001.Legacy.GetPlotMarkerStyle(nPlotID)
```

GetPlotStepMode method

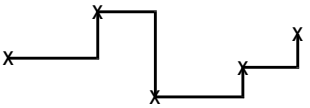
Applicable to

Legacy object

Description

Returns the “step mode” of a plot. The following figures show how step mode affects a plot (the samples are the same in both plots).

A plot that uses step mode



A plot that doesn’t use step mode

Syntax

```
GetPlotStepMode plotID
```

Part	Description
plotID	The plot’s ID.

Return values

The step mode, which can be:

`1` = Step mode is used

`0` = Step mode is not used

Remarks

- Step mode makes it easier for operators to use the ToolTip functionality to see a sample value. (If step mode is not used, the ToolTip shows the interpolated value.)

Example

This example gets the step mode setting for the current plot.

```
Dim nPlotID, bStepMode
nPlotID = chart001.Legacy.CurrentPlot
bStepMode = chart001.Legacy.GetPlotStepMode(nPlotID)
```

GetPlotVisible method

Applicable to

Legacy object

Description

Returns the visibility setting of a plot.

Syntax

```
GetPlotVisible plotID
```

Part	Description
plotID	The plot's ID.

Return values

The plot's visibility setting, which can be:

0 = Hidden

1 = Visible

Example

This example shows how to add a new plot to chart003, and then check its visibility setting. (Note that "g_PlotID1" is a global page variable that used in later scripts to identify the plot.)

This adds the plot using the AddNewPlot method.

```
g_PlotID1 = chart003.AddNewPlot("Tank1 level", 0, 100, "Meters",
   XRangeMin, XRangeMax, "Time")
```

Later scripts can check the plot's visibility setting.

```
Dim visible
visible = chart001.Legacy.GetPlotVisible(g_PlotID1)
  If visible = 0 Then
    .
    .
  Else
    .
    .
  End If
```

GetRange method

Applicable to

XAxis object

YAxis object

Description

Returns the current range displayed on the axis.

Syntax

```
GetRange (varLow, varHigh)
```

Part	Description
varLow	The low range value.
varHigh	The high range value.

Return values

The range on the axis.

GetReferenceValues method

Applicable to

Legacy object

ReferenceCursor object

Description

Returns an array of reference values and the corresponding dates/times for each plot intersected by the reference line. (See “Reference line and reference values” on page 224.)

Syntax

```
GetReferenceValues plotIDs, refvalues, refdatetimes
```

Part	Description
plotID	The plot IDs.
refvalues *refdatetimes*	The plot reference value and date/time for the plots.

Return values

None.

Remarks

- The first plot (plot ID = 0) is the default plot and always exists—even if no plots have been added to the trend—and it is used to show default range information. If no plots have been added to the trend, the current plot is set to the default plot. (That is, the trend’s CurrentPlot property is set to 0.)

Example

This example displays the reference values (and associated plot IDs and dates/times) in a set of textboxes (textbox001 to textbox*nnn*) when the user creates a reference line (clicks in the plot view).

```
Sub customtrend001_OnReferenceLineSet(bSet)
  If bSet = 1 Then
    Dim plotIDs(), refvaluess(), refdatetimes()

    customtrend001GetReferenceValues plotIDs, refvaluess,
   refdateTimes

    textbox001.value = plotIDs(1)
    textbox002.value = refvaluess(1)
    textbox003.value = refdatetimes(1)
```

```
        .
        .
      textboxnnn.value = refdatetimes(x)
    End If
  End Sub
```

GetResponse method

Applicable to

Application object

Station object

Description

Displays the specified prompt in Station's Message Zone and returns the response entered by the user in Station's Command Zone.

Syntax

```
GetResponse prompt
```

Part	Description
prompt	The message that appears in the Message Zone.

Return values

The user's response.

Remarks

- Station cannot perform any other task while waiting for a response from the user. You should therefore only use GetResponse within a suitable timer subroutine, as shown in the following example.

Example

This example shows how to use a timer-initiated subroutine to check for a response, and if there is none (indicated by a null string) exit the routine.

This sets up the timer (in this example, on a button onclick event).

```
Sub StartBtn_onclick()
  m_iTimerID = window.setInterval ("TimerHandler()", 1000)
End Sub
```

This is the timer-initiated subroutine.

```
Sub TimerHandler()
  Dim str
  str = window.external.Parent.GetResponse ("Are you sure?")
    If str = "" then
      'Haven't received a response yet
      .
      .
    Else
      'Received a response! Now let's process it.  First kill
   the timer.
```

```
            window.clearInterval(m_iTimerID)
            .
            .
        End If
    End Sub
```

GetResponse2 method

Applicable to

Application object

Description

Displays the specified prompt in Station's Message Zone. The OnResponse event fires when the user responds to the prompt.

Syntax

```
GetResponse2 prompt
```

Part	Description
prompt	The message that appears in the Message Zone

Return values

None.

Remarks

- Do not use this method for display-level scripting because Station events cannot be caught by page scripts. You can, however, use the method in an SSO, as shown in the following example.

Example

This example calls the GetResponse2 method in the SSO's OnBeforeConnect event handler and saves the result. When the onresponse event is fired, the connection to the cached connection file is either made or rejected.

```
Private Sub m_objStation_OnBeforeConnect(ByVal strConnectionFile
   As String, ByRef bCancel As Boolean)
  If m_strChangeConnection = "n" Then
    m_objStation.GetResponse2 "Are you sure you want to change
   connections (y/n)?"
    bCancel = True
    'Save the connection file so that we can connect later if
   the user says yes
    m_strConnectionFile = strConnectionFile
  Else
    bCancel = False
  End If
End Sub
```

The onresponse event is then fired when the user response is received.

```
Private Sub m_objStation_onresponse()
  m_strChangeConnection = window.event.getAttribute
   ("OnResponse", 0)
  If m_strChangeConnection = "n" And !IsEmpty(m_
   strConnectionFile) Then
    m_objStation.Connect m_strConnectionFile
  End If
End Sub
```

GetResponse3 method

Applicable to

Application object

Description

Displays the specified prompt in Station's Message Zone and returns the user's response entered in Station's Command Zone.

Syntax

```
GetResponse3 prompt, type, response
```

Part	Description
prompt	The message that appears in the Message Zone.
type	The prompt type: `0` = Presents the user with an input box, and **OK** and **Cancel** buttons. `1` = Presents the user with **Yes** and **No** buttons.
response	The user's response string. If the prompt type is `1`, the user response string will be either a `y` for yes, or `n` for no.

Return values

The response status, which can be:

`0` = The response has been canceled.

`1` = The response is valid. Note that the response has to be checked to determine whether the response has been received. If the response is a null string (" "), then the response has not been received.

Remarks

- For a display-level script, you should only use this method within a suitable timer subroutine, as shown in the following example.
- This method is not available in eServer premium.

Example

This example shows how to set up a timer-initiated subroutine. The subroutine is called repeatedly at each timer interval until the user responds.

This script uses the button's onclick event to set up a 1000 ms timer.

```
Sub StartBtn_onclick()
  m_iTimerID = window.setInterval ("TimerHandler()", 1000)
End Sub
```

This is the timer-initiated subroutine that obtains the user's response. If there is no response (indicated by a null string), the subroutine terminates without killing the timer.

```
Sub TimerHandler()
  Dim str
  Dim iStatus
  Dim ptPromptType = 1
  iStatus = window.external.GetResponse3 ("Are you sure?",
   ptPromptType, str)
    If iStatus = 1 then
      If str = "" then
        'Haven't received a response yet
        .
        .
      Else
        'Received a response! Now let's process it.
        .
        .
      End If
    Else
      'The response has been canceled so kill the timer so we
   don't poll for a response again
    window.clearInterval(m_iTimerID)
      .
      .
  End If
End Sub
```

GetResponse4 method

Applicable to

Application object

Description

Displays the specified prompt in Station's Message Zone and returns the user's response entered in Station's Command Zone.

Syntax

```
GetResponse3 prompt, type, response
```

Part	Description
prompt	The message that appears in the Message Zone.
type	The prompt type: `0` = Presents the user with an input box, and **OK** and **Cancel** buttons. `1` = Presents the user with **Yes** and **No** buttons.
response	The user's response string. If the prompt type is `1`, the user response string will be either a `y` for yes, or `n` for no.

Return values

None.

Remarks

- If you use this method in an SSO, you can use the OnResponse event to process the user's response.
- This method is not available in eServer premium.

GetSampleInterval method

Applicable to

Trend object

Description

Returns the current sample interval information.

Syntax

```
GetSampleInterval (nHours, nMinutes, nSeconds, nMilliseconds,
bAverage)
```

Part	Description
nHours	The number of hours in the sample.
nMinutes	The number of minutes in the sample.
nSeconds	The number of seconds in the sample.
nMilliseconds	The number of milliseconds in the sample.
bAverage	Whether the sample is an average or a snapshot.

Return values

The number of time units in the sample.

The sample type, which can be:

`True` = Average

`False` = Snapshot

GetStyleProperty method

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Group object

Indicator object

Page object

Picture object

Pushbutton object

Shape object

Text object

Vector Graphic object

Description

Returns the value of a style attribute in the style associated with the object. (To learn about styles, see “About style sheets” on page 108.)

Syntax

```
GetStyleProperty styleattribute
```

Part	Description
styleattribute	The name of the style attribute. See “Predefined style attributes” on page 114.

Return values

The value of the style attribute.

If the style attribute is not defined in the style or no style is associated with the object, `vbNull` is returned. The following example shows how to check whether `vbNull` is returned.

Remarks

- Use the GetStyleClassProperty method to obtain the value of a style attribute in specific style.

Example

This example obtains the color defined for “hw-fill-color” in the style attached to “rect001”.

```
if rect001.GetStyleProperty("hw-fill-color") <> vbNull then
  textbox001.value = rect001.GetStyleProperty("hw-fill-color")
end if
```

GetStyleClassProperty method

Applicable to

Page object

Description

Returns the value of a style attribute in a specific style. (To learn about styles, see "About style sheets" on page 108.)

Syntax

```
GetCustomProperty style, styleattribute
```

Part	Description
style	The name of the style.
styleattribute	The name of the style attribute. See "Predefined style attributes" on page 114.

Return values

The value of the style attribute in the specified style.

If the style attribute is not defined in the style or if the style does not exist, `vbNull` is returned. The following example shows how to check whether `vbNull` is returned.

Remarks

- Use the GetStyleProperty method to obtain the value of a style attribute associated with a particular object (or the display's background).

Example

This example obtains the color defined for "hw-fill-gradient-color" in the style, "IndicatorType1".

```
if page.GetStyleClassProperty("IndicatorType1", "hw-fill-
   gradient-color") <> vbNull then
  color = page.GetStyleClassProperty("IndicatorType1", "hw-fill-
   gradient-color")
end if
```

GetValue method

Applicable to

StationDataRepository object

DisplayDataRepository object

Description

Returns the value of a system custom property in the Station data repository.

Syntax

```
GetValue name
```

Part	Description
name	The name of the custom property

Return values

The value of the custom property specified by the name.

GetXAxisToolbarItemVisible method

Applicable to

ChartArea object

Description

Returns the visibility of the x-axis toolbar buttons.

Syntax

```
GetXAxisToolbarItemVisible (itemID, pvVisible)
```

Part	Description
itemID	The ID of the button on the toolbar.
pvVisible	Returns the visibility of the button.

Return values

The visibility of the x-axis toolbar which can be:

0 = Hidden

1 = Visible

Item ID values, which can be:

0 = PAUSE

1 = RESUME

2 = RESET_ZOOMING

3 = ZOOM_IN

4 = ZOOM_OUT

5 = SHOW_TIME_SELECTOR

6 = REMOVE_REFERENCE_LINE

GetXAxisScale method

Applicable to

Plots object

Description

Returns the visible range on the x-axis.

Syntax

```
GetXAxisVisibleRange (varXLow, varXHigh)
```

Part	Description
varXLow	The left-most visible point on the x-axis.
varxHigh	The right-most visible point on the x-axis.

Return values

The left- and right-most visible point on the x-axis.

GetYAxisScale method

Applicable to

Plot object

Description

Returns the visible range of the y-axis for the plot.

Syntax

```
GetYAxisVisibleRange (pvarYLow, pvarYHigh)
```

Part	Description
pvarYLow	The bottom-most visible point on the y-axis.
pvarYHigh	The top-most visible point on the y-axis.

Return values

The top- and bottom-most visible point on the y-axis.

GetXAxisUnzoomedScale

Applicable to

Plots object

Description

Returns the visible range on the x-axis scale when the trend is not zoomed.

Syntax

```
GetXAxisUnzoomedScale (varXLow, varXHigh)
```

Part	Description
varXLow	The left-most visible point on the x-axis.
varxHigh	The right-most visible point on the x-axis.

Return values

The left- and right-most visible point on the x-axis.

Remarks

- If the trend is zoomed and an operator zooms out, the trend returns to this scale.

GetYAxisUnzoomedScale

Applicable to

Plot object

Plots object

Description

Returns the visible range on the y-axis scale when the trend is not zoomed.

Syntax

```
GetYAxisUnzoomedScale (varYLow, varYHigh)
```

Part	Description
pvarYLow	The bottom-most visible point on the y-axis.
pvarYHigh	The top-most visible point on the y-axis.

Return values

The top- and bottom-most visible point on the y-axis

Remarks

- If the trend is zoomed and an operator zooms out, the trend returns to this scale.

GetZoomRegion method

Applicable to

PlotCanvas object

Description

Returns the zoomed area of the chart.

Syntax

```
GetZoomRegion (nXStart, nXEnd, nYStart, nYEnd)
```

Part	Description
nXStart	The start point on the x-axis of the zoomed area.
nXEnd	The end point on the x-axis of the zoomed area.
nYStart	The start point on the y-axis of the zoomed area.
nYEnd	The end point on the y-axis of the zoomed area.

Return values

The start and end points of the x and y-axis of the zoomed area.

Green method

Applicable to

Application object

Description

The Green method is a utility function that can be used in any script. It returns the green component of a color value expressed as a number between 0 (no green) and 255 (100% green). Normally used in conjunction with the Blue and Red methods.

Syntax

```
window.external.Green RGBvalue
```

Part	Description
RGBvalue	The color value from which the green component is returned

Return values

The green component of the color.

Example

This example obtains the green component of the fill color of “Rect3”.

```
GreenValue = window.external.Green(Rect3.FillColor)
```

HideAllDataTips method

Applicable to

EventBar object

Description

Hides all visible event data tips.

Syntax

```
HideAllDataTips
```

Return values

None.

HideDataTips method

Applicable to

EventBar object

Description

Hides event data tips for the specified events.

Syntax

```
HideDataTips varEventIDList
```

Part	Description
verEventIDList	The list of event IDs.

Return values

None.

IncreaseZoomLevel method

Applicable to

PlotCanvas object

Description

Increases the zoom by approximately 20%.

Syntax

```
IncreaseZoomLevel
```

Return values

None.

InvokeCommand method

Applicable to

Application object

Station object

Description

Executes the specified command as if it had been selected by the user from a menu or toolbar.

Syntax

```
InvokeCommand command
```

Part	Description
command	The command that is executed. The list of commands is shown in Station's **Customize** dialog box. To see the list, choose **Station** > **Connection Properties**, click the Toolbars tab and then click **Customize**.

Return values

None.

Remarks

- This method is not available in eServer premium.

Example

This example shows how to call up the point detail display (for the current point).

```
window.external.Parent.InvokeCommand "Request Point Detail"
```

InvokeMenu2 method

Applicable to

Application object

Description

Invokes a command associated with a menu item or toolbar button as if it had been chosen/clicked by the user.

Syntax

```
InvokeMenu2 commandname
```

Part	Description
commandname	The name of the command, as specified in Station's **Customize** dialog box. See "Finding the name of a Station command" on page 128.

Return Values

None.

Remarks

- This method is not available in eServer premium.

Example

This example shows how to invoke the command that is executed when the user clicks on Station's toolbar. (This calls up the point detail display, and the command's name is "Request Point Detail".)

```
window.external.InvokeMenu2 "Request Point Detail"
```

InvokePopup method

Applicable to

Application object

Station object

Description

Calls up a popup or faceplate as if it had been called up by the user.

Syntax

```
InvokePopup filename, Xpos, Ypos, popuptype
```

Part	Description
filename	The filename of the popup/faceplate.
Xpos *Ypos*	The x- and y-coordinates of the top-left of the popup relative to the parent display. (The coordinates of the top-left of the display are `1 1`.) If you specify `0 0`, Station uses the default the positioning rules for popups (even if it is a faceplate). This is generally below and to the right of the parent object.
popuptype	The type of popup: `0` = Popup `1` = Faceplate

Return Values

None.

Remarks

- This method is not available in eServer premium.

Example

This example shows how to call up a popup called “procedure5.htm”.

```
window.external.Parent.InvokePopup “procedure5.htm”, 100, 100, 0
```

Item method

Applicable to

Dictionary object

StationWindows object

Description

For the Dictionary object, it returns the specified item from the Dictionary.

For the StationWindows object, it returns the display at the index from the StationWindow collection.

Syntax

```
Item index|name
```

Part	Description
index	Returns the item with the specified index (sequence number within the list of items). For the Dictionary object, index numbering starts at 1. For the StationWindows object, index numbering starts at 0.
name	For the Dictionary object, returns the item with the specified name. The name is case-sensitive. For the StationWindows object, returns the display at the index.

Return values

None.

Example

This example returns the value of a Dictionary item using its index.

```
SavedValue = window.external.Dictionary.Item(1).Value
```

This example returns the value of a Dictionary item using its name.

```
SavedValue = window.external.Dictionary.Item("Volts").Value
```

This example changes the fill-color of all display objects on the display.

```
For i = 1 to Page.Objects.Count
  Page.all.Item(i).FillColor = vbRed
Next
```

This example loops through the open displays.

```
Dim i As Integer
For i = 0 To (m_objStation.Parent.StationWindows.Count - 1)
     MsgBox m_
   objStation.Parent.StationWindows.Item(i).CurrentPage
Next
```

KillTimer method

Applicable to

Application object

Station object

Description

Only available to SSOs—see “Writing Station-level scripts” on page 320. Not available to display-level scripts.

Kills the specified timer. Timers are created by the CreateTimer method.

Syntax

```
KillTimer timer
```

Part	Description
timer	The ID of the timer that is killed.

Remarks

- This method is not available in eServer premium.

LogMessage method

Applicable to

Application object

Station object

Description

Adds an entry in the Station log. The Station log, `log.txt`, is located in `C:\Documents and Settings\All Users\Application Data\Honeywell\HMIWebLog`. (Note that, by default, this folder is set as a system folder, which means that it is only visible if you select the **Show hidden files and folders** check box in your Windows Explorer settings.)

Syntax

```
LogMessage message
```

Part	Description
message	The message to be added to the Station log.

Return values

None.

Remarks

- This method is not available in eServer premium.

Example

This example adds an entry to the log.

```
window.external.Parent.LogMessage "Operation complete"
```

LPToDPX and LPToDPY methods

Applicable to

AppWindow object

Description

These two methods convert the X- and Y-coordinates, respectively, from HMIWeb Display Builder's logical coordinates to the resolution-dependent display coordinates (expressed in pixels). See "Specifying dimensions and position" on page 321.

Syntax

```
LPToDPX(logicalcoordx)
LPToDPY(logicalcoordy)
```

Part	Description
logicalcoordx *logicalcoordy*	The logical coordinate that is converted.

Remarks

- The DPToLPX and DPToLPY methods convert from display coordinates to logical coordinates.

Example

This example obtains the display coordinates of "Rect1".

```
Xcoord = window.external.AppWindow.LPToDPX(Rect1.Left)
Ycoord = window.external.AppWindow.LPToDPY(Rect1.Top)
```

MakeColor method

Applicable to

Application object

Description

The MakeColor method is a utility function that can be used in any script. It returns a color value, expressed in RGB format. The value is packed into the lower 24 bits of a long integer.

Syntax

```
MakeColor red, green, blue
```

Part	Description
red	The red component, with a value between 0 (no red) and 255 (maximum red).
green	The green component, with a value between 0 (no green) and 255 (maximum green).
blue	The blue component, with a value between 0 (no blue) and 255 (maximum blue).

Return values

None.

Example

This example changes the fill color of "Rect3" to white.

```
Rect3.FillColor = window.external.MakeColor(255,255,255)
```

MessageBox method

Applicable to

Station object

Description

Creates a message box which is a child of Station.

Syntax

```
MessageBox text, buttoncode
```

Part	Description
text	The text that appears in the message box.
buttoncode	Specifies which button(s) appear in the message box: • MB_OK • MB_OKCANCEL • MB_ABORTRETRYIGNORE • MB_YESNOCANCEL • MB_YESNO • MB_RETRYCANCEL

Return values

The code indicating which button was clicked:

1 = OK

2 = Cancel

3 = Abort

4 = Retry

5 = Ignore

6 = Yes

7 = No

Remarks

- If you are creating an SSO, you should use this method in preference to the VBScript MsgBox function to ensure that the message box is visible to the Station user.

Objects method

Applicable to

Shape object

Description

Obtains the child object of a shape with a specified name.

Syntax

```
Objects name
```

Part	Description
name	The name of the child object.

Return values

The shape's child object with the specified name.

Remarks

- If the script is within a display, you must use this method to reference the children of a shape. However, if the script is within the shape file, you can directly reference the objects within the shape by name.

Example

This example changes the color of the vector object called "rect1" contained within "shape1" to red.

```
shape1.Objects("rect1").fillColor = vbRed
```

PutValue method

Applicable to

StationDataRepository object

DisplayDataRepository object

Description

Sets the value of a system custom property in the Station data repository.

Syntax

```
PutValue name
```

Part	Description
name	The name of the system custom property

Return values

None.

Example

This example sets the value of the system custom property with a scope of Station "Custom1" to "tank1".

```
window.external.StationDataRepository.PutValue("Custom1"
   "tank1")
```

This example sets the value of the system custom property with a scope of display "Custom1" to "tank1".

```
DisplayDataRepository.PutValue("Custom1" "tank1")
```

Quit method

Applicable to

Application object

Station object

Description

Terminates Station. Equivalent to the user choosing **File > Exit**.

Syntax

```
Quit
```

Return values

None.

Remarks

- The method does not prompt for user confirmation.

Example

This example performs shut-down tasks and terminates the Station.

```
Sub QuitButton_OnClick()
  'Perform shut-down tasks.
  .
  .
  window.external.Parent.Quit
End Sub
```

RaiseCustomEvent method

Applicable To

Application object

Description

Calling this method will fire an event that will be received by all implementors of the oncustomevent event. This could include page scripts (on all pages in a multi-window system), application scripts, external objects using Station's automation interface and external objects using the Desktop Framework Event Model (DFEM) event system.

This is a system developed to allow Honeywell applications to exchange events. Using the DFEM component an application can raise and receive events across applications. Station integrates with this system and allows page and application scripts to fire events using RaiseCustomEvent and to receive events using OnCustomEvent.

However, the most common of these uses by display developers is to fire events between displays in a multi-window system. When any display script calls window.external.RaiseCustomEvent all display page objects will receive an OnCustomEvent and can use the passes parameters to determine if the event is relevant to them.

Syntax

```
RaiseCustomEvent(bstrNamespace, bstrURN, svarEventObject)
```

Part	Description
bstrNamespace	The name of the group of events this event belongs to.
bstrURN	The name of the event
svarEventObject	The value of an object related to the event. The value can be of any form, such as text or numeric.

Red method

Applicable to

Application object

Description

The Red method is a utility function that can be used in any script. It returns the red component of a color value, expressed as a number between 0 (no red) and 255 (100% red). Normally used in conjunction with the Blue and Green methods.

Syntax

```
Red RGBvalue
```

Part	Description
RGBvalue	The color value from which the red component is returned.

Return values

The red component of the color.

Example

This example obtains the red component of the fill color of “Rect3”.

```
RedValue = window.external.Red(Rect3.FillColor)
```

Redraw method

Applicable to

Legacy object

Description

Redraws the chart.

Syntax

```
Redraw redrawarea
```

Part	Description
redrawarea	The area which is redrawn: 1 = Plot view only 2 = X axis view only 4 = Y axis view only 16 = Y scrollbar only 32 = X scrollbar only 64 = Unzoom button only 127 = Redraw everything

Return values

None.

Example

This example shows how to add a new plot to chart003, and then redraw it. (Note that "g_PlotID1" is a global page variable that used in later scripts to identify the plot.)

This adds the plot using the AddNewPlot method.

```
g_PlotID1 = chart003.Legacy.AddNewPlot("Tank1 level", 0, 100,
   "Meters", XRangeMin, XRangeMax, "Time")
```

This redraws the plot.

```
dim PLOTVIEW_ONLY
PLOTVIEW_ONLY = 1
If alpha001.value > 80 Then
  chart003.Legacy.SetPlotColor g_PlotID1, vbRed
  'Redraw plot view only
  chart003.Legacy.Redraw(PLOTVIEW_ONLY)
End If
```

Refresh method

Applicable to

Application object

Description

Refreshes the display, which includes obtaining the latest database values from the server.

Syntax

```
Refresh
```

Return values

None.

Remarks

- Use this method in preference to the standard DHTML reload method because it does not necessarily update database values.

ReloadContent method

Applicable to

Shape object

Description

Reloads the content of the shape if its autoReloadContent property is set to `False`.

Syntax

```
ReloadContent
```

Return values

None.

Example scenarios

Changing a trend's contents at run time

Remove method

Applicable to

Dictionary object

StationDataRepository object

DisplayDataRepository object

Description

Removes an item from the object.

Syntax

```
Remove index|name
```

Part	Description
index	The sequence number of the item that is removed. Index numbering starts at 1.
name	The name of the item that is removed. The name is case-sensitive.

Return values

None.

Example

This example removes an item called “Voltage” from the Dictionary.

```
window.external.Dictionary.Remove "Voltage"
```

This example removes the third item from the Dictionary.

```
window.external.Dictionary.Remove 3
```

This example removes the system custom property “Custom1” from the Station data repository.

```
window.external.StationDataRepository.Remove “Custom1”
```

This example removes the system custom property “Custom1” from the display data repository.

```
DisplayDataRepository.Remove “Custom1”
```

RemovePlot method

Applicable to

Legacy object

Description

Removes a plot from the chart.

Syntax

```
RemovePlot plotID
```

Part	Description
plotID	The plot's ID.

Return values

None.

Example

This example shows how to add a new plot to chart003, and then remove it. (Note that "g_PlotID1" is a global page variable that used in later scripts to identify the plot.)

This adds the plot using the AddNewPlot method.

```
g_PlotID1 = chart003.Legacy.AddNewPlot("Tank1 level", 0, 100,
   "Meters", XRangeMin, XRangeMax, "Time")
```

Later scripts can remove the plot.

```
chart003.Legacy.RemovePlot g_PlotID1
```

Remarks

- If you want to retain the plot but clear the current data, use the ClearPlotData method.

RemovePlotData method

Applicable to

Legacy object

Description

Clears all or some of the plot sample data associated with a plot. All other plot information, such as the range, remains the same.

Syntax

```
RemovePlotData plotID, plotxdata, optimizedflag
```

Part	Description
plotID	The plot's ID.
plotxdata	The x-axis values (date/time) whose corresponding y-axis values are removed.
optimizedflag	Set to 0. (Reserved for future functionality.)

Return values

None.

RequestDemandData method

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Hyperlink object

Indicator object

Picture object

Pushbutton object

Shape object

Text object

Vector Graphic object

Description

Requests updated values for all "on-demand" point parameters associated with an object. (These are point parameters listed on the object's Script Data tab, whose **Update rate** is set to `On Demand`.)

Syntax

```
RequestDemandData
```

Return values

None.

Remarks

- An ondatachange event is fired when the updated value(s) are received. However, because this event is also fired as a result of other updates (such as a Station update), you may want to use the object's DataChanged property to check whether the update was the result of the RequestDemandData call.

  (If an attempt is made to obtain an "on-demand" value before it has been updated, the value obtained by the last RequestDemandData call will be returned. However, if the first RequestDemandData call has not yet completed, the value will be undefined except for the QualityBad property, which will return `True`.)

Example

This example shows how to request an update for on-demand parameters associated with "rect001".

This code shows how to request the update.

```
Sub pushbutton001_onclick
  rect001.RequestDemandData
End Sub
```

This code shows how to use the object's DataChanged property to check whether the update was the result of the RequestDemandData call.

```
Sub rect001_ondatachange
  if (rect001.DataChanged("On Demand")) then
    'Code to execute if True
    .
    .
  end if
End Sub
```

RequestServerLRN method

Applicable to

Application object

Station object

Description

Requests the server task with the associated LRN (Logical Resource Number).

Syntax

```
RequestServerLRN LRN [, param1, param2]
```

Part	Description
LRN	The requested LRN.
param1 *param2*	The numeric parameter(s) passed to the LRN. These are specific to the task and action being requested.

Return values

None.

Remarks

- For more information about the Server Display Program (LRN 21) actions and parameters, see the *Configuration Guide*.
- For more information about displaying the LRN assigned to an application program, see the *Application Development Guide*.
- If you need to pass a string to the task, such as the name of a display or a point, use RequestTask method.

Example

This example uses the Server Display Program (LRN 21) to call up the numbered DSP display, whose page number is 11.

```
window.external.Parent.RequestServerLRN 21, 1, 11
```

RequestTask method

Applicable to

Application object

Station object

Description

Requests the server task with the associated LRN (Logical Resource Number), with the specified parameters.

Syntax

```
RequestTask LRN [, param1, param2, param3, param4, paramDetails]
```

Part	Description
LRN	The requested LRN.
param1 *param2* *param3* *param4*	The numeric parameter(s) passed to the LRN. These are specific to the task and action being requested.
paramDetails	Only applicable when calling the Server Display Program (LRN 21). Passes a string that would normally be entered by the user in Station's Command Zone if the function was performed manually, for example the name of a display or the name of a point.

Return values

None.

Remarks

- This method is similar to RequestServerLRN method; however, it allows greater flexibility when used with the Server Display Program (LRN 21) because it can pass a string to the task, such as the name of a display or a point.
- For more information about the Server Display Program (LRN 21) actions and parameters, see the *Configuration Guide*.
- For more information about displaying the LRN assigned to an application program, see the *Application Development Guide*.

Example

This example uses the Server Display Program (LRN 21) to call up a HMIWeb display called “testPageCallup.htm”.

```
window.external.Parent.RequestTask 21, 72, 0, 0, 0,
   "testPageCallup.htm"
```

This example uses the Server Display Program (LRN 21) to call up the point detail display for a point called “tank5”.

```
window.external.Parent.RequestTask 21, 73, 0, 0, 0, "tank5"
```

ResetContent method

Applicable to

Combobox object

Description

Removes all items from the combobox's list.

Syntax

```
ResetContent
```

Return values

None.

Example

This example removes all items from the combobox called "combobox001" when the user clicks pushbutton "pushbutton001".

```
Sub pushbutton001_onclick
  combobox001.ResetContent
End Sub
```

RuntimeDataRequest method

Applicable to

Trend object

Description

Requests data for a trend if the `AutoRuntimeDataRequest` property is set to `False`.

Syntax

```
RuntimeDataRequest
```

Return values

None.

Scroll method

Applicable to

XAxis object

YAxis object

Description

Scrolls the axis back or forward by the specified amount.

Syntax

```
Scroll nScrollAmount
```

Part	Description
nScrollAmount	The amount of scrolling of the axis. A positive amount scrolls the axis forward, a negative amount scrolls the axis back.

Return values

None.

SendKeyPress method

Applicable to

Application object

Description

Simulates a key being pressed—that is, as if the key was pressed by the user.

Syntax

```
SendKeyPress keycode
```

Part	Description
keycode	The key code of the pressed key.

Return values

None.

Remarks

- This method is not available in eServer premium.

SetCommandUIState method

Applicable to

Station object

Description

Enables or disables the specified command. When a command is disabled:

- The associated toolbar button disappears
- The associated menu item is grayed out
- The associated keyboard short cut is disabled
- Scripts cannot invoke the command

Syntax

```
SetCommandUIState commandID, state
```

Part	Description
commandID	The command's ID, as defined in the Station's toolbar, menu and keyboard shortcut definition file (`*.stb`).
state	The command's state: `0` = Disabled `1` = Enabled

Return values

`0` if successful.

Example

This script prevents users from changing Station's connection.

```
window.external.Parent.SetCommandUIState "Open Connection", 0
```

SetCustomProperty method

Applicable to

Shape object

Description

Sets the value of the specified custom property.

Syntax

```
SetCustomProperty type, name, value
```

Part	Description
type	The custom property's type.
name	The custom property's name.
value	The custom property's value.

Return values

None.

Remarks

- This method operates asynchronously, so after calling the method, it is not possible to reference the subcontent of the shape object. If you want to do this, you must create a script in the onshapeload event of the shape to perform any operations that need to reference the subcontent of the shape object.
- If you want to set several custom properties, you can improve the performance by first setting the shape's autoReloadContent property to False, then setting the custom properties, and finally using the ReloadContent method. For an example of this technique, see the script in "Changing a trend's contents at run time" on page 671.
- To learn about custom properties, see "About custom properties" on page 57.
- Use the GetCustomProperty method to get value of a custom property.

Example scenarios

Changing a trend's contents at run time

SetDataTipArrowColors method

Applicable to

EventBar object

Description

Sets the color of the arrows associated with multiple events in an event data tip.

Syntax

```
SetDataTipArrowColors (clrArrow, clrButtonFace)
```

Part	Description
pclrArrow	The arrow used in data tips
pclrButtonFace	The background on which the arrow appears.

Return values

None.

SetDataTipFormat method

Applicable to

ReferenceCursor object

Description

Sets the format of the data tip currently used for the reference cursor.

Syntax

```
SetDataTipFormat (ctType, strFormat)
```

Part	Description
ctType	The chart type.
strFormat	Specifies what information is shown when the mouse hovers above the reference cursor.

Return values

The chart type, which can be:

1 = Single bar chart

2 = Single scale line chart

3 = Line chart

The information shown in the data tip, which can be:

%x = X-axis value

%X = X-axis units

%y = Y-axis value

%Y = Y-axis units

Remarks

The default formats of the data tip are:

Single bar chart = %x %X

Single scale line chart = %x %X

Line chart = %y %Y %x

SetDataTipVisible method

Applicable to

EventBar object

Description

Sets the visibility of data tips for the specified event.

Syntax

```
SetDataTipVisibility (nEventID, bVisible)
```

Part	Description
nEventID	The ID of the event you want to show or hide.
bVisible	The event visibility setting:

Return values

The visibility of data tips, which can be:

0 = Hidden

1 = Visible

SetGridLinePen method

Applicable to

PlotCanvas object

Description

Sets the line-related properties for the gridlines in the plot area.

Syntax

```
SetGridLinePen (nLineStyle, nLineWidth, clrLine)
```

Part	Description
nLineStyle	The line style of the gridlines: • 0 = Solid line • 1 = Dashed line • 2 = Dotted line • 3 = Dashed/dotted line • 4 = Dashed/double-dotted line
nLineWidth	The width of the gridline (in pixels).
clrLine	The color of the gridline.

Return values

None.

Remarks

- This method is ignored if the GridLineAutoColor property is set to `True`.

SetInPlaceEditing method

Applicable to

XAxis object

YAxis object

Description

For the x-axis, enables or disables the date/time control and sets the position of the time control.

For the y-axis, enables or disables scaling of the axis.

Syntax

```
SetInPlaceEditing (varbEnabled, nPosition)
```

Part	Description
varbEnabled	Sets whether editing is enabled.
nPosition	Sets the position of the date/time control.

Return values

The position state, which can be:

0 = SHOW_RIGHT

0 = SHOW_TOP

1 = SHOW_LEFT

1 = SHOW_BOTTOM

2 = SHOW_CENTRE

3 = SHOW_LEFT_RIGHT

3 = SHOW_TOP_BOTTOM

4 = HIDDEN

setInterval method

Applicable to

window object

Description

Evaluates an expression each time a specified number of milliseconds has elapsed.

Syntax

```
iTimerID=window.setInterval(vcode,iMilliseconds [, sLanguage])
```

Part	Description
vcode	A required string that specifies the function pointer or string that indicates the code to be executed when the specified level has elapsed.
iMilliseconds	A required integer that specifies the number of milliseconds.
sLanguage	A optional string that specifies the language used. The values that can be used are: • `Jscript` - Language is JScript • `VBScript` - Language is VBScript • `JavaScript` - Language is JavaScript

Return values

A integer that is used as an identifier that cancels the timer with the clearInterval method. See “clearInterval method” on page 433.

Remarks

- The setInterval method continuously evaluates the specified expression until the timer is removed with the clearInterval method.
- For more information on timers see “Timers” on page 291.

Example

- See “Creating an animation with a shape sequence” on page 650.

SetItemVisible method

Applicable to

ChartToolbar object

TitleToolbar object

Description

Sets the visibility of items in the chart toolbar.

Syntax

```
SetItemVisible (tbItem, vbItemVisible)
```

Part	Description
tbItem	The toolbar item you want to show or hide, which can be: • 1 = The View trend drop-down button and the Show legend button. • 2 = The View trend drop-down button • 3 = The Show legend button • 20 = The y-axis label, which indicates the units for the currently selected plot. • 21 = This includes: – Plot color indicator – Plot selection combobox – Scale configuration dropdown button – Chart type dropdown button • 22 = This includes the time period combo and the interval combo. • 23 = The scale configuration dropdown button. • 24 = The chart type selection dropdown button.
vbItemVisible	The visibility settings: • 0 = Hidden • 1 = Visible

Return values

None.

SetLabels method

Applicable to

Plot object

Description

Sets labels for the plot, x-axis, and y-axis.

Syntax

```
SetLabels (strPlotLabel, strXLabel, strYLabel)
```

Part	Description
strPlotLabel	The plot label.
strXLabel	The x-axis label.
strYLabel	The y-axis label.

Return values

None.

SetLineStyle method

Applicable to

ReferenceCursor object

Description

Sets the line style of the reference cursor.

Syntax

```
SetLineStyle (clrLine, nWidth)
```

Part	Description
clrLine	The color of the reference cursor.
nWidth	The width of the reference cursor.

Return values

None.

Remarks

- The reference cursor is solid if the width is greater than 1 pixel.

SetNodePoint method

Applicable to

Node object

Description

Sets the x- and y-coordinates of a node.

Syntax

```
SetNodePoint x, y
```

Part	Description
x *y*	The node's x-and y-coordinates.

Return values

None.

Example

This example sets the first node of a polyline.

```
Dim objNode
set objNode = polyline001.Nodes.Item(0)
objNode.SetNodePoint 10, 20
```

SetPen method

Applicable to

Plot object

Description

Sets the line-related properties of a plot.

Syntax

```
SetPen (nLineStyle, nLineWidth, clrLine)
```

Part	Description
nLineStyle	The line style of the plot: • 0 = Solid line • 1 = Dashed line • 2 = Dotted line • 3 = Dashed/dotted line • 4 = Dashed/double-dotted line
nLineWidth	The width of the plot line (in pixels).
clrLine	The color of the plot.

Return values

None.

SetPeriod method

Applicable to

Trend object

Description

Sets the period visible within the chart area.

Syntax

```
SetPeriod (nYears, nMonths, nWeeks, nDays, nHours, nMinutes,
nSeconds)
```

Part	Description
nYears	The number of years in the period.
nMonths	The number of months in the period.
nWeeks	The number of weeks in the period.
nDays	The number of days in the period.
nHours	The number of hours in the period.
nMinutes	The number of minutes in the period.
nSeconds	The number of seconds in the period.

Returns values

None.

SetPlotColor method

Applicable to

Legacy object

Description

Sets the color of a plot.

Syntax

```
SetPlotColor plotID, linecolor
```

Part	Description
plotID	The plot's ID.
linecolor	The color of the plot. See "Setting and returning colors" on page 303.

Return values

None.

Example

This example shows how to add a new plot to chart003, and then set its color. (Note that "g_PlotID1" is a global page variable that used in later scripts to identify the plot.)

This adds the plot using the AddNewPlot method.

```
g_PlotID1 = chart003.Legacy.AddNewPlot("Tank1 level", 0, 100,
   "Meters", XRangeMin, XRangeMax, "Time")
```

Later scripts can set the plot's color.

```
chart003.Legacy.SetPlotColor g_PlotID1, vbRed
```

SetPlotDiscrete method

Applicable to

Legacy object

Description

Sets the plot's "line style". (That is, whether or not lines appear between samples.)

Syntax

```
SetPlotDiscrete plotID, linestyle
```

Part	Description
plotID	The plot's ID.
linestyle	The line style: • 0 = Lines between samples • 1 = Markers only (If no marker style has been specified, each sample will be rendered as an "x".)

Return values

None.

Example

This example checks whether lines appear between samples in the current plot, and removes them if they are currently shown.

```
Dim nPlotID, bDiscrete
nPlotID = chart001.Legacy.CurrentPlot
chart001.Legacy.GetPlotDiscrete nPlotID, bDiscrete
if bDiscrete = 0 Then
  Chart001.Legacy.SetPlotDiscrete nPlotID, 1
End if
```

SetPlotMarkerStyle method

Applicable to

Legacy object

Description

Sets the plot's marker style. (The symbol used to mark samples in the chart.)

Syntax

```
SetPlotMarkerStyle plotID, markerstyle
```

Part	Description
plotID	The plot's ID.
markerstyle	The plot's marker style, which can be: • 0 = None • 1 = X • 2 = Plus • 3 = Square • 4 = Diamond • 5 = Triangle • 6 = DownTriangle • 7 = Circle • 19 = SquareSolid • 20 = DiamondSolid • 21 = TriangleSoild • 22 = DownTriangleSoild • 23 = CircleSolid

Example

This example sets the marker style of the current plot to circle.

```
Dim nPlotID
nPlotID = chart001.Legacy.CurrentPlot
chart001.Legacy.SetPlotMarkerStyle nPlotID, 7
```

SetPlotPen method

Applicable to

Legacy object

Description

Sets the line-related properties of a plot.

Syntax

```
SetPlotPen plotID, linestyle, linewidth, linecolor
```

Part	Description
plotID	The plot's ID.
linestyle	The plot's line style: • 0 = Solid line • 1 = Dashed line • 2 = Dotted line • 3 = Dashed/dotted line • 4 = Dashed/double-dotted line **Note:** Line styles other than solid, are not supported if *linewidth* is greater than one pixel.
linewidth	The width of the plot line (in pixels).
linecolor	The color of the plot. See "Setting and returning colors" on page 303.

Return values

None.

Example

This example shows how to add a new plot to chart003, and then set its line-related properties. (Note that "g_PlotID1" is a global page variable that used in later scripts to identify the plot.)

This adds the plot using the AddNewPlot method.

```
g_PlotID1 = chart003.Legacy.AddNewPlot("Tank1 level", 0, 100,
   "Meters", XRangeMin, XRangeMax, "Time")
```

Later scripts can set the plot's line-related properties.

```
chart003.Legacy.SetPlotPen g_PlotID1, 1, 1, vbGreen
```

SetPlotRange method

Applicable to

Legacy object

Description

Sets a plot's range. The plot range is the default range of the data that is shown. This can also be thought of as the range/area of data shown when the chart is zoomed out to 100%.

Syntax

```
SetPlotRange plotID, ylow, yhigh, xlow, xhigh
```

Part	Description
plotID	The plot's ID.
ylow *yhigh*	The y-axis low and high limits.
xlow *xhigh*	The x-axis low and high limits.

Return values

None.

Remarks

- Do not set the x-axis range if the data is related to a point. (Using a script to change the range would cause the display to be out of sync with the server—which would cause unpredictable behavior when scrolling the chart.)

  If you want to change the y-axis range for point-related data, use the GetPlotRange method. For example:

```
Dim YLow, YHigh, XLow, XHigh
Chart001.Legacy.GetPlotRange chart001.currentPlot, YLow, YHigh,
   XLow, XHigh
Chart001.Legacy.SetPlotRange chart001.currentPlot, 20, 80, XLow,
   XHigh
```

Example

This example shows how to add a new plot to chart003, and then set its range. (Note that "g_PlotID1" is a global page variable that used in later scripts to identify the plot.)

This adds the plot using the AddNewPlot method.

```
g_PlotID1 = chart003.Legacy.AddNewPlot("Tank1 level", 0, 100,
   "Meters", XRangeMin, XRangeMax, "Time")
```

Later scripts can set the plot's range.

```
chart003.Legacy.SetPlotRange g_PlotID1, 20, 80, #20/5/2002
   12:00:30 PM#, #20/5/2002 12:01:00 PM#
```

SetPlotStepMode method

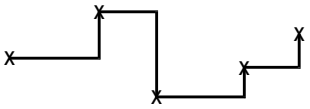
Applicable to

Legacy object

Description

Sets the "step mode" of a plot. The following figures show how step mode affects a plot (the samples are the same in both plots).

A plot that uses step mode



A plot that doesn't use step mode

Syntax

```
GetPlotStepMode plotID, mode
```

Part	Description
plotID	The plot's ID.
mode	The step mode: • 1 = Step mode enabled • 0 = Step mode disabled

Remarks

- Step mode makes it easier for operators to use the ToolTip functionality to see a sample value. (If step mode is not used, the ToolTip shows the interpolated value.)

Example

This example turns on step mode for the current plot.

```
Dim nPlotID
nPlotID = chart001.Legacy.CurrentPlot
chart001.Legacy.setPlotStepMode nPlotID, 1
```

SetPlotVisible method

Applicable to

Legacy object

Description

Shows or hides a plot.

Syntax

```
SetPlotVisible plotID, visibility
```

Part	Description
plotID	The plot's ID.
visibility	The plot's visibility setting: • 0 = Hidden • 1 = Visible

Return values

None.

Example

This example shows how to add a new plot to chart003, and then set its visibility. (Note that "g_PlotID1" is a global page variable that used in later scripts to identify the plot.)

This adds the plot using the AddNewPlot method.

```
g_PlotID1 = chart003.Legacy.AddNewPlot("Tank1 level", 0, 100,
   "Meters", XRangeMin, XRangeMax, "Time")
```

Later scripts can control the plot's visibility.

```
chart003.Legacy.SetPlotVisible g_PlotID1, 0
```

SetSampleInterval method

Applicable to

Trend object

Description

Sets the sample interval and sample type for the trend.

Syntax

```
SetSampleInterval (nHours, nMinutes, nSeconds, nMilliseconds,
bAverage)
```

Part	Description
nHours	The number of hours in the sample.
nMinutes	The number of minutes in the sample.
nSeconds	The number of seconds in the sample.
nMilliseconds	The number of milliseconds in the sample.
bAverage	Sets whether the sample is an average or a snapshot: • `True` = average • `False` = snapshot

Return values

None.

SetSampleIntervaltoFastHistory method

Applicable to

Trend object

Description

Sets the fast history interval to match the fast history interval as configured for the Experion server.

Syntax

```
SetSampltIntervaltoFastHistory
```

Return values

None.

setTimeout method

Applicable to

window object

Description

Evaluates an expression after a specified number of milliseconds has elapsed.

Syntax

```
iTimerID=window.setTimeout(vcode,iMilliseconds [, sLanguage])
```

Part	Description
vcode	A required string that specifies the function pointer or string that indicates the code to be executed when the specified level has elapsed.
iMilliseconds	A required integer that specifies the number of milliseconds.
sLanguage	A optional string that specifies the language used. The values that can be used are: • `Jscript` - Language is JScript • `VBScript` - Language is VBScript • `JavaScript` - Language is JavaScript

Return values

A integer that is used as an identifier that cancels the evaluation with the clearTimeout method. See “clearTimeout method” on page 437.

Remarks

- The specified expression or function is evaluated once.
- For more information on timers see “Timers” on page 291.

SetTimeRange method

Applicable to

Trend object

Description

Sets the time range for the x-axis.

Syntax

```
SetTimeRange (dtLeftHandTime, dtCentreTime, dtRightHandTime)
```

Part	Description
dtLeftHandTime	Sets the time on the left time control.
dtCentreTime	Sets the time on the centre time control.
dtRightHandTime	Sets the time on the right time control.

Return values

None.

Remarks

- If you want to set or change the time on the left time control only and leave the period unchanged, set *dtLeftHandTime* to the required value and set *dtCentreTime* and *dtRightHandTime* to 0.
- If you want to set or change the time on the center time control only and leave the period unchanged, set *dtCentreHandTime* to the required value and set *dtLeftHandTime* and *dtRightHandTime* to 0.
- If you want to set or change the period, set *dtLeftHandTime* and *dtRightHandTime* to the required value and set *dtCentreTime* 0.

SetTitleBarFormat method

Applicable to

Station object

Description

Sets the content of Station's title bar.

Syntax

```
SetTitleBar string
```

Part	Description
string	The string that appears in the title bar. The string can include the following variables: • %C = The Station setup file, for example `default.stn` • %T = The title of the current display (or the display's filename if it does not have a title) • %N = The display's filename (or number if it is a *numbered* display) • %P = The name of the point (only valid for a point detail display) You can include optional items using %[*x*...], where *x* can be C, T, N or P. For example, %[P : %N] would result in " : sysDtlAna" if the display (point detail) was empty; otherwise it would result an empty string.

Return values

None.

Remarks

- Station's default title is: "%C - %T%[N(%N)]%[P : %P]". For example, if a display numbered (80) and its title is "Home" the title would be something like:

  ```
  Station - Default.stn - Home (80)
  ```

  For the point detail display for an analog point called "SINEWAVE", the title would be something like:

  ```
  Station - Default.stn - Analog Point Detail(sysDtlAna) :
  SINEWAVE
  ```

Example

This example configures the title bar so that it only shows the setup file.

```
window.external.Parent.SetTitleBarFormat "%C"
```

SetUserVersionInfo method

Applicable to

Station object

Description

Adds a string to the information provided in the Station's About box. Developers of external components (like Station Scripting Objects) can display version information within Station.

Syntax

```
SetUserVersionInfo informationstring
```

Part	Description
informationstring	The string displayed below the Station version information in the Station's About box.

SetXAxisToolbarItemVisible method

Applicable to

ChartArea object

Description

Sets the visibility of buttons on the x-axis toolbar.

Syntax

```
SetXAxisToolbarItemVisible (itemID, bVisible)
```

Part	Description
itemID	The toolbar button ID: • 0 = Pause • 1 = Resume • 2 = Reset zooming • 3 = Zoom in • 4 = Zoom out • 5 = Show time selector
bVisible	The visibility settings: • 0 = Hidden • 1 = Visible

Return values

None.

SetXAxisScale method

Applicable to

Plots object

Description

Sets the visible range on the x-axis.

Syntax

```
SetXAxisVisibleRange (varXLow varXHigh)
```

Part	Description
varXLow	The x-axis low limit.
varXHigh	The x-axis high limit.

Return values

None.

Remarks

- If your Station and server are in different timezones, the time value that is set will be different from what you expect, the difference being the amount of time difference between the Station and server.

SetYAxisScale method

Applicable to

Plot object

Plots object

Description

Sets the visible range on the y-axis.

Syntax

```
SetYAxisScale (varYLow, varYHigh)
```

Part	Description
varYLow	The bottom-most visible point on the y-axis.
varYHigh	The top-most visible point on the y-axis.

Return values

None.

SetXAxisUnzoomedScale method

Applicable

Plots object

Description

Sets the visible range on the x-axis when there is no zooming on the trend. If the trend is zoomed and an operator zooms out, the trend returns to this scale.

Syntax

```
SetXAxisUnzoomedScale (varXLow, varXHigh)
```

Part	Description
varXLow	Sets the left-most visible point on the x-axis.
varXHigh	Sets the right-most visible point on the x-axis.

Return values

None.

Remarks

- If the trend is zoomed and an operator zooms out, the trend returns to this scale.
- If your Station and server are in different timezones, the time value that is set will be different from what you expect, the difference being the amount of time difference between the Station and server.

SetYAxisUnzoomedScale method

Applicable to

Plot object

Plots object

Description

Sets the visible range on the y-axis when there is no zooming on the trend.

Syntax

```
SetYAxisUnzoomedScale (varYLow, varYHigh)
```

Part	Description
varYLow	Sets the bottom-most visible point on the y-axis.
varYHigh	Sets the top-most visible point on the y-axis.

Return values

None.

Remarks

- If the trend is zoomed and an operator zooms out, the trend returns to this scale.

SetZoomRegion method

Applicable to

PlotCanvas object

Description

Sets the zoomed area of the chart.

Syntax

```
SetZoomRegion (nXStart, nXEnd, nYStart, nYEnd)
```

Part	Description
nXStart *nXEnd*	The x-coordinates (horizontal) of the zoom area, where 0 is the left-most point and 1 is the right-most point.
nYStart *nYEnd*	The y-coordinates (vertical) of the zoom area, where 0 is the bottom-most point and 1 is the top-most point.

Return values

None.

Shell method

Applicable to

Application object

Station object

Description

Executes a command as an operating system command.

Syntax

```
window.external.Parent.Shell command, state
```

or

```
Call window.external.Parent.Shell(command, state)
```

Part	Description
command	The operating system command.
state	Sets the state of the window in which the command runs. The values are: • 0 = Window is hidden (not visible). • 1 = Standard Window • 2 = Window is minimized • 3 = Window is maximized

Return values

None.

Example

This example starts Windows Notepad and opens a text file.

```
window.external.Parent.Shell "notepad c:\documents\readme.txt",1
```

UnZoom method

Applicable to

PlotCanvas object

Legacy object

Description

Resets a chart to its normal scale (that is, removes any zooming effects).

Syntax

```
UnZoom
```

Return values

None.

UserObjectNotify method

Applicable to

Application object

Description

Fires an OnUserObjectNotify event.

This method is used by automation objects that may require significant processing time, thereby avoiding script timeouts.

Syntax

```
UserObjectNotify(lID, vParam)
```

Part	Description
lID	The numeric ID that identifies the object using the method.
vParam	The variant used to return data to the script.

Events

The events are listed in alphabetical order.

Attention

Because of the sheer size and complexity of the DOM, this help only describes events that are either specific to HMIWeb displays, or are particularly important.

onactivate event

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Group object

Hyperlink object

Indicator object

Page object

Picture object

Pushbutton object

Shape object

Text object

Vector Graphic object

Description

Standard DHTML event that fires when the object is set as the active element.

Remarks

- Bubbles and cannot be canceled.
- The complementary event is the ondeactivate event.

OnActivated event

Applicable to

Application object

Page object

Description

Fires when Station becomes the active application.

OnAlarm event

Applicable to

Application object

Page object

Station object

Description

Fires when a new alarm appears in the Station’s Alarm Zone.

OnAppStartup event

Applicable to

Application object

Station object

Description

Fires as soon as Station is started and all its objects have been created.

Remarks

- The event is not fired when the Station script is reloaded, for example, when the user chooses **Station** > **Connect**.

OnAppStatusUpdate event

Applicable to

Application object

Page object

Station object

Description

Fires when there is a change in status of one or more boxes in Station's Status Line.

Syntax

For the Application and Station objects:

```
OnAppStatusUpdate(changedfields)
```

Part	Description
changedfields	An array, which specifies the status changes. See "Status Line boxes" on page 396.

For the Page object:

```
OnAppStatusUpdate()
```

In the case of the Page object, you must use the event object to explicitly retrieve the status changes. For example:

```
stationstatus = window.event.getAttribute ("OnAppStatusUpdate",
   0)
```

The retrieved value is an array which specifies the status changes. See "Status Line boxes" on page 396.

OnBeforeConnect event

Applicable to

Application object

Page object

Station object

Description

Fires when Station is about to start, or about to change its connection—for example, when a user selects another Station setup file (`.stn`).

Syntax

For the Application and Station objects:

```
OnBeforeConnect(stnfile, connectionallowed)
```

Part	Description
stnfile	The filename (and path) of the Station setup file.
connectionallowed	Indicates whether the Station is allowed to change the connection. The value is either `True` (change connection) or `False` (cancel the change).

For the Page object:

```
OnBeforeConnect()
```

In the case of the Page object, you must use the event object to explicitly retrieve the name (and path) of the setup file. For example:

```
strConnectionFile = window.event.getAttribute
   ("OnBeforeConnect", 0)
```

You can cancel the connection using the event object's returnValue property, for example:

```
strConnectionFile = event.returnValue = False
```

Remarks

- This event can be used in conjunction with the GetConnectionProperty method to check whether the connection properties specified in the setup file are appropriate before it is used by the Connect method.

OnBeforeOKBKeyPressed event

Applicable to

Application object

Station object

Description

Fires when the user presses a key on an Operator Entry Panel (OEP) or Integrated Operator Keyboard (IKB).

The event also specifies whether the key is processed by Station.

Syntax

```
OnBeforeOKBKeyPressed(keycode, cancelpress)
```

Part	Description
keycode	The keycode of the pressed key.
cancelpress	Indicates whether the key function is processed by Station. The value is either `True` (function associated with key executes) or `False` (function associated with key doesn't execute).

Remarks

- This event allows an external application to react when an operator key is pressed and also to cancel this key *before* being processed by Station.
- See also the OnSilenceIKB event and OnOKBKeyPressed event.

OnBlink event

Applicable to

Application object

Page object

Station object

Description

Fires when Station's blink state changes. (It blinks approximately two times a second.) This event allows synchronous blinking between Station and custom displays and/or custom display elements.

Syntax

For the Application and Station objects:

```
OnBlink(blinkstate)
```

Part	Description
blinkstate	The blink state, either `True` (on) or `False` (off).

For the Page object:

```
OnBlink()
```

In the case of the Page object, you must use the event object to explicitly retrieve the blink state. For example:

```
blinkstate = window.event.getAttribute ("OnBlink", 0)
```

The retrieved value is `True` if blinking is on; otherwise `False`.

onblur event

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Group object

Hyperlink object

Indicator object

Page object

Picture object

Pushbutton object

Shape object

Text object

Vector Graphic object

Description

Standard DHTML event that fires when the object loses the input focus.

Remarks

- Can be invoked with the blur method.
- The event does not bubble (see “Event bubbling” on page 287).

onchange event

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Hyperlink object

Indicator object

Picture object

Pushbutton object

Shape object

Text object

Vector Graphic object

Description

Standard DHTML event that fires when the object's value changes.

Remarks

- When the onchange event fires for an object, the new value can be obtained via the Value property. The previous value is obtained via the currentValue property.
- The ondatachange event fires when there is a change to the object's point-related parameters.
- The event does not bubble (see "Event bubbling" on page 287).

Example

```
Sub alpha1_onchange
  If alpha1.value > 10 Then
    window.external.MessageZoneText = "Can't increase
   temperature by more than 10 degrees"
     alpha1.CancelChange
  End If
End Sub
```

onclick event

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Group object

Hyperlink object

Indicator object

Page object

Picture object

Pushbutton object

Shape object

Text object

Vector Graphic object

Description

Standard DHTML event that fires when the user clicks the left mouse button on the object.

Remarks

- Can be invoked programmatically with the click method.

Example scenarios

Using a button to start a pump

Showing and hiding objects

OnConnect event

Applicable to

Application object

Page object

Station object

Description

Fires when Station connects to a server.

Example

This example adds the named variable `myKey` to the Dictionary, with value `10`.

```
Sub Application_OnConnect()
  Window.External.Dictionary.Add "myKey", 10
End Sub
```

oncontextmenu event

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Group object

Hyperlink object

Indicator object

Page object

Picture object

Pushbutton object

Shape object

Text object

Vector Graphic object

Description

Standard DHTML event that fires when the user right-clicks the object.

Remarks

If you have invoked a shortcut menu with a script on the oncontextmenu event of a group, then you need to add the line:

```
window.event.cancelBubble=true
```

after calling InvokeShortcutMenu. Otherwise the default system shortcut menu will appear overriding the menu bought up via a script.

OnCustomEvent event

Applicable to

Application object

Page object

Station object

Description

Fires when an custom event is raised. The custom event could be raised by a page script calling RaiseCustomEvent (in a multi-window system this could be on another page), an external object calling Application.RaiseCustomEvent or another application or page using the Desktop Framework Event Model (DFEM) event system.

This is a system developed to allow Honeywell applications to exchange events. Using the DFEM component an application can raise and receive events across applications. Station integrates with this system and allows page and application scripts to fire events using RaiseCustomEvent and to receive events using OnCustomEvent.

However, the most common of these uses by display developers is to fire events between displays in a multi-window system. When any display script calls window.external.RaiseCustomEvent all display page objects will receive an OnCustomEvent and can use the passes parameters to determine if the event is relevant to them.

Note that by default the following Station events are fired into the DFEM event system with the namespace “HSC_Station_Application”:

- OnAlarm event
- OnOperatorChange event
- OnPageLoad event
- OnPeriodicUpdate event
- OnQuit event
- OnTimer event
- OnUserObjectNotify event
- OnConnect event
- OnDisconnect event
- OnActivated event
- OnOperatorCommand2 event
- OnBeforeConnect event

- OnBlink event
- OnAppStatusUpdate event
- OnDictionaryValueChanged event

To ensure that you don't use these events in your OnCustomEvent handler check the namespace that the incoming event uses. For example to only respond to events from a specific namespace use code like:

```
if (window.event.getAttribute("OnCustomEvent_NameSpace", 0) =
   "test") then
msgbox window.event.getAttribute("OnCustomEvent_EventObject", 0)
end if
```

Syntax

For the Application and Station objects:

```
OnCustomEvent(bstrNamespace, bstrURN, svarEventObject)
```

Part	Description
bstrNamespace	The name of the group of events this event belongs to.
bstrURN	The name of the event.
svarEventObject	The value of an object passed by the event firer. The value can be of any form, such as text or numeric.

For the Page object:

```
OnCustomEvent
```

In the case of the Page object, you must use the event object to explicitly retrieve the following parameters. For example:

```
strCommand = window.event.getAttribute "OnCustomEvent", 0
```

Parameter	Description
OnCustomEvent_URN	The name of the group of events this event belongs to.
OnCustomEvent_NameSpace	The name of the event.
OnCustomEvent_EventObject	The value of an object passed by the event firer. The value can be of any form, such as text or numeric.

ondatachange event

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Group object

Hyperlink object

Indicator object

Picture object

Pushbutton object

Shape object

Text object

Vector Graphic object

Description

Fires when there is a change in the value of a point-related property. (That is, a change in the value of one or more point parameters listed in the object's Script Data tab.)

Remarks

- **Do not** use the ondatachange event to write to a parameter because it may result in an infinite loop when a new ondatachange event fires as a result of your new value. For an example of correct usage, see "Using a button to start a pump" on page 659.
- For a description of the point-related properties, see "Properties and methods of points on the Script Data tab" on page 306.
- The onchange event fires when there is a change to the object's other properties, such as value and fillColor.
- Because all points may not be updated at the same time, you can use the DataChanged property to check if the points with the specified update type have been updated, as shown in the following example.

Example

This code shows how to use the object's DataChanged property to check whether the update was the result of an "On Demand" update.

```
Sub rect001_ondatachange
  if (rect001.DataChanged("On Demand")) or
   (rect001.DataChanged("SINEWAVE.PV")) then
    'Code to execute if True
    .
    .
  end if
End Sub
```

ondblclick event

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Group object

Hyperlink object

Indicator object

Page object

Picture object

Pushbutton object

Shape object

Text object

Vector Graphic object

Description

Standard DHTML event that fires when the user clicks the left mouse button on the object.

ondeactivate event

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Group object

Hyperlink object

Indicator object

Page object

Picture object

Pushbutton object

Shape object

Text object

Vector Graphic object

Description

Standard DHTML event that fires when the activeElement is changed from the current object to another object in the parent document.

Remarks

- Bubbles and cannot be canceled.
- The complementary event is the onactivate event.

OnDictionaryValueChanged event

Applicable to

Application object

Page object

Station object

Description

Fires when an item in the dictionary changes.

Syntax

For the Application and Station objects:

```
OnDictionaryValueChanged(varItemName, varItemValue)
```

Part	Description
varItemName	The name of the item that changed in the dictionary. The name is case-sensitive.
varItemValue	The value of the item that changed in the dictionary. The value can be of any form, such as text or numeric.

For the Page object:

```
OnDictionaryValueChanged()
```

In the case of the Page object, you must use the event object to explicitly retrieve the dictionary item name or values. For example:

```
strItemName = window.event.getAttribute
   ("OnDictionaryValueChanged_ItemName", 0)
strItemValue = window.event.getAttribute
   ("OnDictionaryValueChanged_ItemValue", 0)
```

Part	Description
`OnDictionaryValueChanged_ItemName`	Returns the name of the item that changed in the dictionary.
`OnDictionaryValueChanged_ItemValue`	Returns the value of the item that changed in the dictionary.

OnDisconnect event

Applicable to

Application object

Page object

Station object

Description

Fires when Station disconnects from a server.

Example

This example removes the named variable `myKey` from the Dictionary.

```
Sub Application_OnDisconnect()
  Window.External.Dictionary.Remove("myKey")
End Sub
```

OnDisplayNotFound event

Applicable to

Application object

Station object

Description

Fires if Station cannot find a display it has been asked to load. This event fires before the standard “display not found” error message appears in the Message Zone. The handler of this event can be set to suppress Station’s error message about the display.

Syntax

```
OnDisplayNotFound(displayname, cancelerrormessage)
```

Part	Description
displayname	The name of the display. For point detail displays (and other parameterized displays), the event returns the full URL.
cancelerrormessage	Indicates whether to cancel the error message. The value is either `True` (cancel) or `False` (do not cancel, the default).

Remarks

- The event allows an external application to react when Station is unable to load a specified page.
- This event is only applicable to SSOs and external applications.

Example

This example shows how to use the event, in which “bstrDisplayName” shows the requested display, and “bCancelErrorMessage” is set to `True` if you don’t want Station to show the standard error message.

```
Private Sub m_objStation_OnDisplayNotFound(ByVal bstrDisplayName
   As String, ByRef bCancelErrorMessage As Boolean)
  Dim eMsgResult As VbMsgBoxResult
  eMsgResult = MsgBox("The file " + bstrDisplayName + " cannot
   be found." + vbCrLf + "Should Station show an error
   message?", vbYesNo, "Show Error Message in Station")
  If eMsgResult = vbYes Then
    bCancelErrorMessage = False
  Else
    bCancelErrorMessage = True
  End If
End Sub
```

OnEventDataTipDisplayed event

Applicable to

Trend object

Description

Fires whenever an event data tip has been displayed when events are viewed with the trend.

Syntax

```
OnEventDataTipDisplayed (nEventID, bDisplayed)
```

Part	Description
nEventID	The ID of the event for which the data tips is visible or hidden.
bDisplayed	Returns the visibility of the data tips: • `True` = visible • `False` = hidden

onfocus event

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Group object

Hyperlink object

Indicator object

Page object

Picture object

Pushbutton object

Shape object

Text object

Vector Graphic object

Description

Standard DHTML event that fires when the object gains focus.

Remarks

- Can be invoked programmatically with the focus method.
- The event does not bubble (see “Event bubbling” on page 287).
- We recommend that you don’t use this method as it can interfere with the standard HMIWeb operation. We suggest that you use an onactivate or onclick event instead. See “onactivate event” on page 564 or “onclick event” on page 574.

onkeydown event

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Group object

Hyperlink object

Indicator object

Page object

Picture object

Pushbutton object

Shape object

Text object

Vector Graphic object

Description

Standard DHTML event that fires when the user presses a key.

Remarks

- If you want to detect when the user presses an alphanumeric key, use the onkeypress event.
- The complementary event is the onkeyup event.

onkeypress event

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Group object

Hyperlink object

Indicator object

Page object

Picture object

Pushbutton object

Shape object

Text object

Vector Graphic object

Description

Standard DHTML event that fires when the user presses an alphanumeric key.

Remarks

- If you want to detect when the user presses any key, use the onkeydown event.

onkeyup event

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Group object

Hyperlink object

Indicator object

Page object

Picture object

Pushbutton object

Shape object

Text object

Vector Graphic object

Description

Standard DHTML event that fires when the user releases a key.

Remarks

- Bubbles and cannot be canceled.
- The complementary event is the onkeydown event.

OnMenu event

Applicable to

Application object

Station object

Description

Superseded by the OnMenu2 event.

However the OnMenu event is still supported for scripts on DSP displays that use the InvokeMenu method (described in the *Display Building Guide*).

Syntax

```
OnMenu(item)
```

Part	Description
item	The ID of the selected menu item.

Example

Assume the InvokeMenu method has been used with menu item `2113` (alarm acknowledge). This fires an OnMenu event, which returns `2113`.

The following script displays a message if the menu item is 2113.

```
Sub Application_OnMenu(lMenuTag)
  If cint(lMenuTag) = 2113 Then
    window.external.MessageZoneText = "Warning: Alarm
   acknowledge"
  End if
End Sub
```

OnMenu2 event

Applicable to

Application object

Page object

Description

Fires when the user chooses a menu item.

Syntax

For the Application object:

```
OnMenu2(commandname)
```

Part	Description
commandname	The name of the command, as specified in Station's **Customize** dialog box. See "Finding the name of a Station command" on page 128.

For the Page object:

```
OnMenu2()
```

In the case of the Page object, you must use the event object to explicitly retrieve the command. For example:

```
strCommandID = window.event.getAttribute ("OnMenu2", 0)
```

The command names are shown in Station's **Customize** dialog box. See "Finding the name of a Station command" on page 128.

Remarks

- This event supersedes the OnMenu event.
- If this event is associated with a faceplate, the faceplate needs to have focus.

Example

This example for handling the event in a display script displays a warning when the user chooses **Action > Acknowledge/Silence**. (The name of associated command is "Acknowledge Alarm".)

```
Sub Application_onmenu2()
  strCommandID = window.event.getAttribute ("OnMenu2", 0)
  If strCommandID = "Acknowledge Alarm" Then
     window.external.MessageZoneText = "WARNING: Alarm
   acknowledged"
  End If
End Sub
```

onmousedown event

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Group object

Hyperlink object

Indicator object

Page object

Picture object

Pushbutton object

Shape object

Text object

Vector Graphic object

Description

Standard DHTML event that fires when the user clicks the object with either mouse button.

Remarks

- Does not bubble and cannot be canceled.
- The complementary event is the onmouseup event.

onmouseenter event

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Group object

Hyperlink object

Indicator object

Page object

Picture object

Pushbutton object

Shape object

Text object

Vector Graphic object

Description

Standard DHTML event that fires when the mouse pointer enters the object's bounds.

Remarks

- Does not bubble and cannot be canceled. (Compare with onmouseover event, which does bubble and can be canceled.)
- The complementary event is the onmouseleave event.

onmouseleave event

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Group object

Hyperlink object

Indicator object

Page object

Picture object

Pushbutton object

Shape object

Text object

Vector Graphic object

Description

Standard DHTML event that fires when the mouse pointer leaves the object's bounds.

Remarks

- Does not bubble and cannot be canceled. (Compare with onmouseout event, which does bubble and can be canceled.)
- The complementary event is the onmouseenter event.

onmousemove event

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Group object

Hyperlink object

Indicator object

Page object

Picture object

Pushbutton object

Shape object

Text object

Vector Graphic object

Description

Standard DHTML event that fires when the user moves the mouse over the object.

Remarks

- Does not bubble and cannot be canceled.

onmouseout event

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Group object

Hyperlink object

Indicator object

Page object

Picture object

Pushbutton object

Shape object

Text object

Vector Graphic object

Description

Standard DHTML event that fires when the mouse pointer leaves the object's bounds.

Remarks

- Bubbles and can be canceled. (Compare with onmouseleave event, which does not bubble and cannot be canceled.)
- The complementary event is the onmouseover event.

onmouseover event

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Group object

Hyperlink object

Indicator object

Page object

Picture object

Pushbutton object

Shape object

Text object

Vector Graphic object

Description

Standard DHTML event that fires when the mouse pointer enters the object's bounds.

Remarks

- Bubbles and can be canceled. (Compare with onmouseenter event, which does not bubble and cannot be canceled.)
- The complementary event is the onmouseout event.

onmouseup event

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Group object

Hyperlink object

Indicator object

Page object

Picture object

Pushbutton object

Shape object

Text object

Vector Graphic object

Description

Standard DHTML event that fires when the user releases the mouse button while over the object.

Remarks

- Does not bubble and cannot be canceled.
- The complementary event is the onmousedown event.

OnOKBKeyPressed event

Applicable to

Application object

Station object

Description

Fires when the user presses a key on an Operator Entry Panel (OEP) or Integrated Operator Keyboard (IKB). The keycode is passed and the event handler is informed if Station has handled the key.

Syntax

```
OnOKBKeyPressed(keycode, handled)
```

Part	Description
keycode	The key code of the key that was pressed.
handled	Indicates whether the key is handled by Station—that is, whether it been assigned to a command. The value is `True` if the key is handled, or `False` if it isn't. The key assignments are defined in Station's toolbar definition file (`*.stb`) file.

Remarks

- The event allows an external application to react when on operator key is pressed and also to handle this key before Station processes it.
- See also the OnSilenceIKB event and OnBeforeOKBKeyPressed event.
- This event is only applicable to SSOs and external applications.

OnOperatorChange event

Applicable to

Application object

Page object

Station object

Description

Fires when the user logs on, or when the user changes security levels.

Example

This example shows how to call up the appropriate “home page” when the user logs on. Each display would contain information appropriate to a particular security level.

```
Sub Application_OnOperatorChange()
  If SecurityLevel = 2 Then
    CurrentPage = "MainMenuOperator"
  End If
  If SecurityLevel = 3 Then
    CurrentPage = "http://www.../overview.htm"
  End If
  If SecurityLevel = 4 Then
    CurrentPage = 303
  End If
  If SecurityLevel = 5 Then
    CurrentPage = 304
  End If
End Sub
```

OnOperatorCommand event

Applicable to

Application object

Station object

Description

Fires when the user issues a command from Station's Command Zone.

Syntax

```
OnOperatorCommand(command)
```

Part	Description
command	The command entered by the user.

Remarks

- Use the OnOperatorCommand2 event if you need to check whether the command included a confirmation prompt, such as "Are you sure? (Y/N)".
- The OnOperatorCommand event fires before the command is processed. This allows the event to be used in conjunction with the CancelOperatorCommand method to validate the command before sending it to the server.

Example

This example checks the user's security level and cancels the command if the level is below Engineer (level 4 security).

```
Sub Application_OnOperatorCommand(bstrCommand)
  If SecurityLevel < 4 Then
    window.external.MessageZoneText = "Only an Engineer or
   Manager can do this"
  CancelOperatorCommand
  End If
End Sub
```

OnOperatorCommand2 event

Applicable to

Application object

Page object

Description

Fires when the user issues a command from Station's Command Zone.

This event enables you to determine whether the command included a confirmation prompt, such as "Are you sure? (Y/N)". If you don't need this functionality, you can use the OnOperatorCommand event.

Syntax

For the Application object:

```
OnOperatorCommand2(command, promptresponse, cancel)
```

Part	Description
command	The command entered by the user before pressing ENTER.
promptresponse	Indicates whether Station is currently awaiting a response to a prompt (such as control confirmation). It is `True` if awaiting a response; otherwise `False`.
cancel	Allows the script to prevent the command from being handled by Station. Return `True` for this value to prevent Station from handling the command; otherwise `False`.

For the Page object:

```
OnOperatorCommand2()
```

In the case of the Page object, you must use the event object to explicitly retrieve the following parameters. For example:

```
strCommand = window.event.getAttribute ("OnOperatorCommand_ID",
   0)
```

Parameter	Description
`OnOperatorCommand_ID`	Returns the command entered by the user.
`OnOperatorCommand_Prompt`	Returns `True` if in response to a prompt; otherwise `False`.

You can cancel the command using the event object's returnValue property, for example:

```
Event.returnValue = True
```

Example

This example for handling the event in a display script checks the user's security level, and cancels the command if the user does not have a high enough level.

```
Private Sub Application_onoperatorcommand2()
strCommand = window.event.getAttribute ("OnOperatorCommand_ID",
   0)
bPromptResponse = window.event.getAttribute ("OnOperatorCommand_
   Prompt", 0)
If SecurityLevel < 4 And bPromptResponse = False Then
    window.external.MessageZoneText = "Only an Engineer or
   Manager can do this"
    Event.returnValue = True
End If
End Sub
```

onoperatorkey event

Applicable to

Page object

Shape object

Description

Fires when the user presses a function key on an Integrated Keyboard (IKB) keyboard or Operator Entry Panel (OEP), but before Station handles it.

Remarks

- You use the event object and its keyCode property to determine which key was pressed.
- You can cancel the key's action by setting the returnValue property of the event object to `False`.
- If this event is associated with a faceplate, the faceplate needs to have focus.

Key codes for IKB function keys

Key	Code
User-defined (top, first row, left to right)	
1	1083310081
2	1083375617
3	1083441153
4	1083506689
5	1083572225
6	1083637761
7	1083703297
8	1083768833
9	1083834369
10	1083899905
11	1085079553
12	1085145089
13	1085210625
14	1085276161
15	1085341697

Key	Code
16	1085407233
17	1085472769
18	1085538305
19	1085603841
20	1085669377
User-defined (top, second row, left to right)	
1	1083965441
2	1084030977
3	1084096513
4	1084162049
5	1084227585
6	1084751873
7	1084817409
8	1084882945
9	1084948481
10	1085014017
11	1085734913
12	1085800449
13	1085865985
14	1085931521
15	1085997057
16	1086521345
17	1086586881
18	1086652417
19	1086717953
20	1086783489
User-defined (top, third row, left to right)	
1	1086849025
2	1086914561
3	1086980097
4	1087045633
5	1087111169
6	1087176705
7	1087242241

Key	Code
8	1087307777
9	1087373313
10	1087438849
11	1082720257
12	1082785793
13	1082851329
14	1082916865
15	1082982401
16	1083047937
17	1083113473
18	1083179009
19	1083244545
20	1083310081
User-defined (top, fourth row, left to right)	
1	1087504385
2	1087569921
3	1087635457
4	1087700993
5	1087766529
6	1087832065
7	1087897601
8	1087963137
9	1088028673
10	1088094209
11	1100283905
12	1100349441
13	1100414977
14	1100480513
15	1100546049
16	1100611585
17	1100677121
18	1100742657
19	1100808193

Key	Code
User-defined (bottom-left, first row, left to right)	
1	1100873729
2	1100939265
User-defined (bottom-left, second row, left to right)	
1	1101463553
2	1101529089
User-defined (bottom-left, third row, left to right)	
1	1101594625
2	1101660161
Labeled keys	
PRIOR DISP	1103298561
DISP FWD	1103495169
PAGE FWD	1103626241
GOTO	1102577665
DISP BACK	1103429633
PAGE BACK	1103560705
PRINT DISP	1102053377
HELP	1103167489
ASSOC DISP	1103364097
GROUP	1102315521
DETAIL	1102381057
SCHEM	1102643201
TREND	1102512129
PRINT TREND	1102118913
FAST	1101922305
HOUR AVG	1103233025
CANCL PRINT	1101987841
RECRD	1101856769
SYST STATS	1101725697
CONS STATS	1101791233
SYST MENU	1102184449
PROC NETWK STATS	1104543745
COMM NETWK STATS	1104609281
AM STATS	1104674817

Key	Code
ORG SUMM	1104740353
UNIT ASGN	1104805889
UNIT TREND	1102446593
ALMS SUMM	1103888385
UNIT ALM SUMM	1104150529
ALMN ANNC	1103953921
MSG SUMM	1103822849
MSG CONFM	1104019457
MSG CLEAR	1104084993
ACK	1103691777
SIL	1103757313
MAN	1104216065
LOAD	1102249985
AUTO	1104281601
SP	1104412673
NORM	1104347137
OUT	1104478209
Raise	1099431937
Fast Raise	1099497473
Lower	1099563009
Fast Lower	1099628545

onpagecomplete event

Applicable to

Page object

Description

Fires when the page is completely loaded, and all objects on the page have been provided with their first update.

Example

This example uses the onpagecomplete as the signal to set the initial colors of two alphanumerics on a page based on their initial values.

```
Sub Page_onpagecomplete
  If alpha1.value > 100 Then
    alpha1.fillColor = vbBlue
  End If
  If alpha2.value > 100 Then
    alpha2.fillColor = vbBlue
  End if
End Sub
```

OnPageLoad event

Applicable to

Application object

Station object

Description

Fires when a page is completely loaded. This occurs when all objects have been created and received their first data update.

OnPause event

Applicable to

Trend object

Description

Fires whenever updates to the trend are paused.

OnPeriodicUpdate event

Applicable to

Application object

Page object

Station object

Description

Fires when Station receives a periodic update from the server.

OnPlotDataBufferFull event

Applicable to

Trend object

Description

Fires when the plot's data buffer becomes full. Each plot can hold a maximum of 4065 samples.

Syntax

```
OnPlotDataBufferFull (nPlotID)
```

Part	Description
nPlotID	The ID of the plot with a full data buffer.

Remarks

- When the trend contains Experion server data, the plot data buffer automatically removes samples to allow for new samples to be added.

OnPlotDetailsChanged event

Applicable to

Trend object

Description

Fires whenever the configuration of the specified plot is changed.

Syntax

```
OnPlotDetailsChanged (nChart, nPlotId, varYRangeLow,
varYRangeHigh, varbPlotEnable)
```

Part	Description
nChart	
nPlotId	The plot ID of the plot that has changed.
varYRangeLow	The y-axis low range for the plot.
varYRangeHigh	The y-axis high range for the plot.
varbPlotEnable	Indicates whether the plot is visible. The value is either: • TRUE = Visible • FALSE = Hidden

OnQuit event

Applicable to

Application object

Station object

Description

Fires when the user closes Station. For example, by choosing **Station > Exit**.

Example

This example changes an alphanumeric value when the user exits Station.

```
Sub Application_OnQuit()
  window.external.MessageZoneText = "Have a nice day!"
End Sub
```

OnReferenceCursorSet

Applicable to

Trend object

Description

Fires whenever the reference line is shown or hidden.

Syntax

```
OnReferenceCursorSet (bSet)
```

Part	Description
bSet	Returns the state of the reference line: • `True` = The reference line is set • `False` = The reference line is hidden

OnRequestDataTip event

Applicable to

Trend object

Description

Fires when a data tip is about to be displayed for a plot in the trend.

Syntax

```
OnRequestDataTip (nPlotID, varXVal, strDataTip)
```

Part	Description
nPlotID	The ID of the plot for which the data tip is to be displayed.
varXVal	The x-axis value where the data tip is located.
strDataTip	The text of the data tip.

Remarks

- You can use this event to customize the data tip.

OnRequestEventInfo event

Applicable to

Trend object

Description

Fires when an event data tip is about to be displayed for an event when trends are are viewed with events.

Syntax

```
OnRequestEventInfo (nEventID, strEventInfo)
```

Part	Description
nEventID	The ID of the event for which the data tip is to be displayed.
strEventInfo	The text of the data tip.

Remarks

- You can use this event to customize the event data tip.

OnRequestReferenceDataTip event

Applicable to

Trend object

Description

Fires when a data tip is about to be displayed for the reference line in the trend.

Syntax

```
OnRequestReferenceDataTip (nPlotID, varXVal, strDataTip)
```

Part	Description
nPlotID	The ID of the plot for which the data tip is to be displayed
varXVal	The x-axis value where the data tip is located.
strDataTip	The text of the data tip.

Remarks

- This event can be used to customize the data tip.

OnResponse event

Applicable to

Application object

Page object

Station object

Description

Fires when the user responds to a message in Station's Message Zone after either the GetResponse2 method or GetResponse3 method was called.

Syntax

For the Application and Station objects:

```
onresponse(response)
```

Part	Description
response	The user's response. If the YesNo prompt type was chosen using the GetResponse3 method, the user response string will be either `y` for yes or `n` for no.

For the Page object:

```
onresponse()
```

In the case of the Page object, you must use the event object to explicitly retrieve the user's response. For example:

```
strResponse = window.event.getAttribute ("OnResponse", 0)
```

Example

This example for handling the event in a display script caches the response string and connects Station using the cached connection file.

```
Private Sub m_objStation_onresponse()
  m_strChangeConnection = window.event.getAttribute
   ("OnResponse", 0)
  If m_strChangeConnection = "n" And !IsEmpty(m_
   strConnectionFile) Then
    m_objStation.Connect m_strConnectionFile
  End If
End Sub
```

OnResume

Applicable to

Trend object

Description

Fires whenever live updates to the trend are resumed.

OnScrollXAxis

Applicable to

Trend object

Description

Fires whenever the x-axis is scrolled.

Syntax

```
OnScrollXAxis (varXLeft, varXRight)
```

Part	Description
varXLeft	Specifies the left-most visible data point in the trend after scrolling.
varXRight	Specifies the right-most visible data point in the trend after scrolling.

OnScrollYAxis event

Applicable to

Trend object

Description

Fires whenever the y-axis is scrolled.

Syntax

```
OnScrollYAxis (varYBottom, varYTop)
```

Part	Description
varYBottom	Specifies the bottom-most data point after the y-axis has been scrolled.
varYTop	Specifies the top-most data point after the y-axis has been scrolled.

onshapeload event

Applicable to

Shape object

Description

The event fires after the associated shape is loaded into the display.

Remarks

- The event fires before the objects on the display have received their first update. To access the value of a data-bound element, you need to use the onupdate event of individual objects, or the onpagecomplete event of the display.
- You can only write a script for the event from within the shape file (where, from the point of view of the Script Editor, it is attached to the Page object). However, at runtime, the event is attached to the shape.
- The event does not bubble (see “Event bubbling” on page 287).

Example

This example changes the value property of the shape to display the third shape in the shape sequence after the shape is loaded into the display.

```
Sub Page_onshapeload()
  Me.value = 3
End Sub
```

onshapeunload event

Applicable to

Shape object

Description

The event fires before the associated shape is unloaded from the display.

Remarks

- You can only write a script for the event from within the shape file (where, from the point of view of the Script Editor, it is attached to the Page object). However, at runtime, the event is attached to the shape.
- The event does not bubble (see “Event bubbling” on page 287).

OnSilenceIKB event

Applicable to

Application object

Station object

Description

Fires when the user presses the SILENCE key on an Operator Entry Panel (OEP) or Integrated Operator Keyboard (IKB).

Remarks

- See also the OnOKBKeyPressed event and OnBeforeOKBKeyPressed event.

OnTimer event

Applicable to

Application object

Page object

Station object

Description

Fires at intervals determined by the associated timer. Timers are created using the CreateTimer method. Timers are cancelled using the KillTimer method.

Syntax

For the Application and Station objects:

```
OnTimer(timer)
```

Part	Description
timerid	Returns the ID of the timer.

For the Page object:

```
OnTimer()
```

You must use the event object to explicitly retrieve that name and path of the setup file. For example:

```
nTimerID = window.event.getAttribute ("OnTimer", 0)
```

Remarks

Use the timer id to distinguish between different timers.

OnTimSelectorPositionChanged event

Applicable to

Trend object

Description

Fires whenever the time selector position on the x-axis is changed.

Syntax

```
OnTimeSelectorPositionChanged (nNewPosition)
```

Part	Description
nNewPosition	The position of the time selector, which can be: • 0 = SHOW_RIGHT • 1 = SHOW_LEFT • 2 = SHOW_CENTRE • 3 = SHOW_LEFT_RIGHT • 4 = HIDDEN

OnTrendLoadComplete event

Applicable to

Trend object

Description

Fires when the trend has finished loading and has received data.

onunload event

Applicable to

Page object

Description

Fires when the page is unloaded.

onupdate event

Applicable to

Alphanumeric object

Checkbox object

Combobox object

Group object

Hyperlink object

Indicator object

Picture object

Pushbutton object

Shape object

Text object

Vector Graphic object

Description

Fires when the value of the object is updated from the server.

Remarks

- The event does not bubble (see “Event bubbling” on page 287).

OnUserObjectNotify event

Applicable to

Application object

Page object

Description

Fires when an automation object calls the UserObjectNotify method. (The method is used by automation objects that may require significant processing time, thereby avoiding script timeouts.)

Syntax

For the Application object:

```
OnUserObjectNotify(lID, vParam)
```

Part	Description
lID	The numeric ID of the object that fired the event.
vParam	The variant used to return data to the script.

For the Page object:

```
OnUserObjectNotify()
```

In the case of the Page object, you must use the event object to explicitly retrieve the following parameters. For example:

```
intObjectID = window.event.getAttribute ("OnUserObjectNotify_
   ID", 0)
```

Parameter	Description
`OnUserObjectNotify_ID`	The numeric ID of the object that fired the event.
`OnUserObjectNotify_ Param`	The variant used to return data to the script.

OnUnZoomed event

Applicable to

Trend object

Description

Fires whenever the trend is zoomed out.

Syntax

```
OnUnZoomed (varXZoomLow, varXZoomHigh, varYZoomLow,
varYZoomHigh)
```

Part	Description
varXZoomLow	Returns the left-most visible data in the trend after zooming out.
varXZoomHigh	Returns the right-most visible data in the trend after zooming out.
varYZoomLow	Returns the bottom-most visible data in the trend after zooming out.
varYZoomHigh	Returns the top-most visible data in the trend after zooming out.

OnZoom event

Applicable to

Trend object

Description

Fires whenever the trend is zoomed in.

Syntax

```
OnZoom (varXZoomLow, varXZoomHigh, varYZoomLow, varYZoomHigh)
```

Part	Description
varXZoomLow	Returns the left-most visible data in the trend after zooming in.
varXZoomHigh	Returns the right-most visible data in the trend after zooming in.
varYZoomLow	Returns the bottom-most visible data in the trend after zooming in.
varYZoomHigh	Returns the top-most visible data in the trend after zooming in.

Legacy_OnChartLoadComplete event

Applicable to

Legacy object

Description

Fires when a trend's chart is loaded and contains data.

Syntax

```
Legacy_OnChartLoadComplete(varTime)
```

Part	Description
varTime	The time (x-axis value) of the most recent sample of data that the chart receives in its initial update.

Remarks

- The event does not fire if the trend is populated by a script or any data source other than the server.

Legacy_OnChartUnZoomed event

Applicable to

Legacy object

Description

Fires when the user resets a chart to its normal scale.

Syntax

```
Legacy_OnChartUnZoomed(xlow, xhigh, ylow, yhigh)
```

Part	Description
xlow *xhigh*	The x-axis low and high limits that applied before the chart was zoomed.
ylow *yhigh*	The y-axis low and high limits that applied before the chart was zoomed.

Legacy_OnChartZoom event

Applicable to

Legacy object

Description

Fires when the user zooms in on a specific part of the chart. (See “Zoom function” on page 225.)

Syntax

```
Legacy_OnChartZoom(xlow, xhigh, ylow, yhigh)
```

Part	Description
xlow *xhigh*	The x-coordinates (horizontal) of the area selected by the user.
ylow *yhigh*	The y-coordinates (vertical) of the area selected by the user.

Remarks

- This event replaces the OnChartZoom event.

Legacy_OnChartScroll event

Applicable to

Legacy object

Description

Fires when the user moves one of the chart's scroll boxes.

Syntax

```
Legacy_OnChartScroll(xboxpos, yboxpos)
```

Part	Description
xboxpos	The new x-origin value (the value at the bottom of the chart).
yboxpos	The new y-origin value (the date/time on the left side of the chart).

Remarks

- This event replaces the OnChartScroll event.

Legacy_OnDataBufferFull

Applicable to

Trend object

Description

Fires when the plot's data buffer becomes full. Each plot can hold a maximum of 4065 samples.

Syntax

```
Legacy_OnDataBufferFull (nPlotID)
```

Part	Description
nPlotID	The ID of the plot with a full data buffer.

Remarks

- When the trend contains Experion server data, the plot data buffer automatically removes samples to allow for new samples to be added.

Legacy_OnEventBoxDisplayed event

Applicable to

Trend object

Description

Fires whenever an event data tip has been displayed when viewing events with trends.

Syntax

```
Legacy_OnEventBoxDisplayed (pvarEvents, bDisplayed)
```

Part	Description
pvarEvents	An array of the event IDs contained within the event data tip.
bDisplayed	Returns the visibility of the data tip. • `True` = visible • `False` = hidden

Legacy_OnEventDetailDisplayed event

Applicable to

Trend object

Description

Fires whenever an event data tip has been displayed when viewing events with the trend.

Syntax

```
Legacy_OnEventDetailDisplayed (nEventID, bDisplayed)
```

Part	Description
nEventID	The ID of the event for which the data tip is visible or hidden.
bDisplayed	Returns the visibility of the data tip: • `True` = visible • `False` = hidden

Legacy_OnReferenceLineSet event

Applicable to

Trend object

Description

Fires when the user creates a reference line. (See “Reference line and reference values” on page 224.)

Syntax

```
Legacy_OnReferenceLineSet(set)
```

Part	Description
set	Returns the reference line’s state: • 1 = The reference line is set • 0 = The reference line is removed/unset

Example

This example displays the reference values (and associated plot IDs and dates/times) in a set of textboxes (textbox001 to textboxnnn) when the user creates a reference line (clicks in the plot view).

```
Sub chart001_Legacy_OnReferenceLineSet(bSet)
  If bSet = 1 Then
    Dim plotIDs(), refVals(), dateTimes()

    chart001.Legacy.GetReferenceValues plotIDs, refVals,
   dateTimes

    textbox001.value = plotIDs(1)
    textbox002.value = refVals(1)
    textbox003.value = dateTimes(1)
   .
   .
    textboxnnn.value = dateTimes(x)
  End If
End Sub
```

Remarks

- This event replaces the OnReferenceLineSet event.

Example scenarios

14

These examples contain scripts that perform a wide range of practical tasks.

Notes

- If you are viewing this online, you can copy example code and paste it into your own scripts—see “Copying example scripts” on page 647.
- If you have previously written scripts for the DSP version of Display, see “Scripting differences between HMIWeb and DSP displays” on page 324.
- The display, `Script_examples.htm`, contains example scripts. It is in `Program Files\Honeywell\Experion PKS\Client\HMIWeb Display Builder\Examples`.

Popups and faceplates

Calling up a faceplate on an alarm condition

ToolTips and messages

Changing a ToolTip based on the state of a pump

Animations

Creating an animation with a shape sequence

Creating a moving object

Controlling an object’s color

Controlling the color of an object in a dynamic shape

Using color to reflect the value of a status point

User controls

Using a button to start a pump

Creating a jogging control

Performing an action in response to a right-click

Checking the user's security level

Inter-application scenarios

Starting an application by clicking a button

Controlling Station from another application

Responding to Station events from another application

Forcing a display to use a particular SafeView window

Trend-related scenarios

Changing a trend's contents at run time

Miscellaneous scenarios

Showing and hiding objects

Using event bubbling to reduce script effort

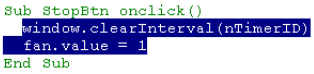
Copying example scripts

If you are viewing this online, you can save a lot of time by copying example scripts, pasting them into your own scripts, and then making the appropriate modifications.

The following example describes how to copy a block of code and paste it into one of your scripts.

To copy and paste a block of code:

1 In the help, drag diagonally across the code that you want to copy.



2 Copy the code by pressing CTRL+C.

3 Switch to the Script Editor.

4 Click where you want to insert the code (typically a blank line), and then press CTRL+V.

Calling up a faceplate on an alarm condition

Scenario

You want to call up a faceplate when a “PV HiHigh” alarm condition occurs on a particular point.

Solution

You add an alphanumeric (or indicator) to the display configure it as follows:

- On the **Data** tab, specify the point ID and set **Parameter** to **AlarmValue**
- On the **Details** tab, set **Display as** to **State Descriptor**
- On the **Behaviors** tab, select **Faceplate**

You write a script that simulates a click (in normal operation, clicking the alphanumeric would call up the faceplate) when the alarm condition occurs. You attach the script to the alphanumeric’s onupdate event.

Scripts

This script simulates an OnClick event when the specified alarm event occurs.

```
Sub alpha002_OnUpdate
  if alpha002.value = "PV HiHigh" Then
    alpha001.click
  End if
End Sub
```

Changing a ToolTip based on the state of a pump

Scenario

You want an object's ToolTip to reflect the state of a pump.

Solution

You write a script that changes the object's title property whenever the state of the pump is open.

Scripts

```
Sub alpha001_onchange
  If alpha001.value = "open" Then
     shape001.title = "Pump is open"
  Else
     shape001.title = "Pump is closed"
  End If
End sub
```

Remarks

- You can create a multi-line ToolTip by inserting `Chr(10)` between each line, for example:

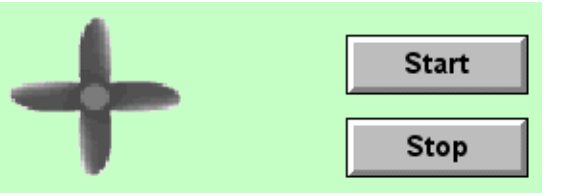
```
shape001.title = "The status of this pump is:" & Chr(10) &
   "CLOSED"
```

Creating an animation with a shape sequence

Scenario

You have a fan that is controlled by **Start** and **Stop** buttons on a display, and want to animate the fan.

The following figure shows the fan and buttons in the display.



Solution

You use the following shape sequence with some scripts to create an animation.

The first (left-hand) shape represents the “off” state, and the other four shapes are used in the animation for the “on” state.

These shapes are bitmap images that were created in a drawing program and then pasted into the shape file. Creating the shapes this way made it easier to produce a more realistic effect.

Scripts

The following script is attached to the **Start** button’s onclick event. This creates a 200 ms timer, which controls the speed of the animation.

```
Sub cmdStartBtn_onclick()
  intTimerID = window.setInterval ("myfunction()", 200)
End Sub
```

The function, “myfunction” is defined in the general section. In this example, the script steps through the four “on” shapes (values 2 to 5).

The general section must also contain the definition of the timer ID variable “intTimerID”, which is used when killing the timer.

```
Dim intTimerID
Function myfunction()
```

```
  If fan.value = fan.shapefile.numberOfShapes Then
    fan.value = 2
  Else
    fan.value = fan.value +1
  End If
End Function
```

The following script is attached to the **Stop** button's onclick event. It stops the animation by calling the clearInterval method of the window object, which takes the timer ID as a parameter. The script also selects the "off" shape.

```
Sub cmdStopBtn_onclick()
  window.clearInterval(intTimerID)
  fan.value = 1
End Sub
```

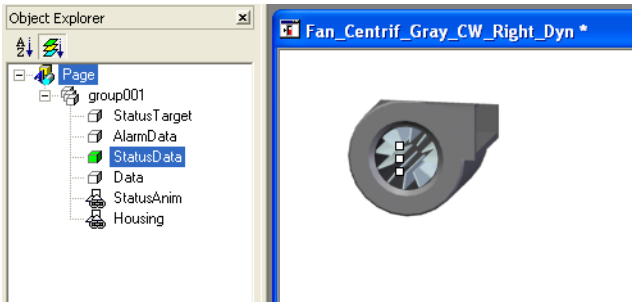
Remarks

- When inserting the shape, you don't have to specify the number of shapes, because this is controlled by your script.

Using timers in a shape file

There are some difficulties in using timers in shape files due to the need to store the timer id as a variable within the scope of the shape. The following scripts provide a solution to overcoming this difficulty.

The script in this example will animate the fan to spin based on the timer value.



The following script is attached into the General section of the display:

```
'Animation Functions
        '--------------------------------------
        sub Animate(oTarget)

        on error resume next

         If oTarget.value = oTarget.shapefile.numberOfShapes
   Then
            oTarget.value = 5
         Else
            oTarget.value = oTarget.value + 1
         End If

        end sub
        '--------------------------------------
```

The following script is attached to the StatusData object and will be run on an onupdate action:

```
Sub StatusData_onupdate

on error resume next

If (me.value = 1) Then
if (not IsNull(group001.parentElement.getAttribute("iTimerID")))
   then
window.clearInterval(group001.parentElement.iTimerID)
group001.parentElement.setAttribute "iTimerID", null
```

```
end if

group001.parentElement.setAttribute "iTimerID",
   window.setInterval("call Animate(StatusAnim)", 100)
Else
if (not IsNull(group001.parentElement.getAttribute("iTimerID")))
   then
window.clearInterval(group001.parentElement.iTimerID)
group001.parentElement.setAttribute "iTimerID", null
end if
StatusAnim.value = 1
End If

StatusTarget.title =
   AlarmData.parentnode.parentnode.GetCustomProperty("value",
   "EquipmentLabel") & "Fan Status is " &
   Data.DataValue("Status.ValueParam")

End Sub
```

Finally, the following code is attached to the AlarmData object to be run on an onupdate:

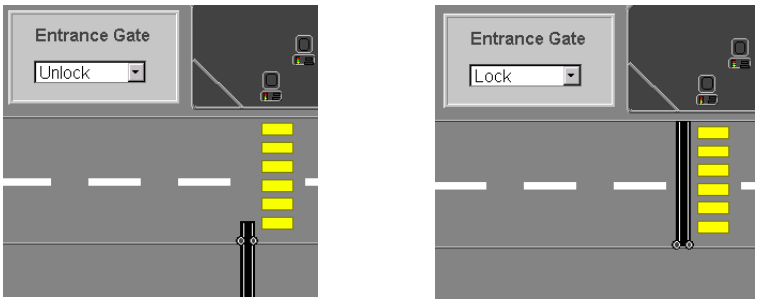
```
Sub AlarmData_onupdate

on error resume next

If me.value = "1" Then
   Housing.title =
   AlarmData.parentnode.parentnode.GetCustomProperty("value",
   "EquipmentLabel") & " has been commanded " &
   Data.DataValue("Start.ValueParam") & " and is in Alarm"
Else
   Housing.title =
   AlarmData.parentnode.parentnode.GetCustomProperty("value",
   "EquipmentLabel") & " has been commanded " &
   Data.DataValue("Start.ValueParam") & " and it's status is " &
   Data.DataValue("Status.ValueParam")
End If

End Sub
```

Creating a moving object

Scenario

You have an object that represents a security gate and want it to “slide” open/closed when you issue an open/close command. (The combobox is used to issue the command.)

Figure 26 The gate, in its open and closed positions



Solution

You create the animation by moving the gate object in small steps at 100 ms intervals. (The step size and interval determine the “smoothness” of the animation.)

A timer is started in the combobox’s onchange event. Each time a timer event fires, the `gateTimer` function moves the gate in the appropriate direction until it is either open or closed.

Scripts

The following script is attached to the combobox’s onchange event. This creates a 100 ms timer, which controls the speed of the animation.

Based on the current state of the combobox, either a `1` or a `0` is written into the variable `blnGateState`.

```
Sub cboGateControl_onchange
  intTimerID=window.setInterval("gateTimer()",100)
  If cboGateControl.value = "On" then
    blnGateState = 1
  Else
    blnGateState = 0
  End If
```

The following script is written into the general section.

The function `gateTimer` is called every 100 ms. It checks the position of the gate by looking at the variable `blnGateState`. If the gate is being opened, it shifts the gate object (`imgGate`) 50 pixels at a time in the open direction until it reaches the fully open position (`600`), and clears the timer. If the gate is being closed, it shifts the gate object 50 pixels at a time in the closed direction until it reaches closed position (`26`), clears the timer and makes sure that the gate is closed in the closed position `26`.

```
Dim blnGateState
Dim intTimerId
Function gateTimer()
  If blnGateState = 1 then
    If imgGate.style.pixelLeft < 600 Then
      imgGate.style.pixelLeft = imgGate.style.pixelLeft + 50
    Else
      window.clearInterval (intTimerId)
    End If
  Else
    If imgGate.style.pixelLeft > 26 then
      imgGate.style.pixelLeft = imgGate.style.pixelLeft - 50
    Else
      window.clearInterval (intTimerId)
      imgGate.style.pixelLeft = 26
    End If
  End If
End Function
```

Controlling the color of an object in a dynamic shape

Scenario

You want to create a dynamic shape to represent a pump. You also want the color of a rectangle to change when the PV of the associated status point changes state.

Solution

When you create the shape, in addition to defining a custom property (see “About custom properties” on page 57), you also:

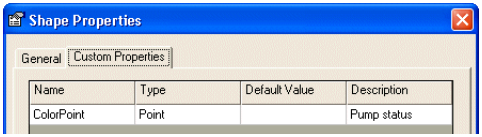
- Add the custom property to the rectangle’s Script Data tab
- Write a script for the rectangle that controls its color based on the value of the custom property

Each time you insert the shape into a display, you assign the custom property to the appropriate point—in effect, the custom property represents the point. Consequently, when the point’s PV changes, so does the rectangle’s color.

To create the shape:

1 Create the new dynamic shape.

2 Open the Properties Window, select the Custom Properties tab and add a custom property, `ColorPoint`, and set its **Type** to `Point`.

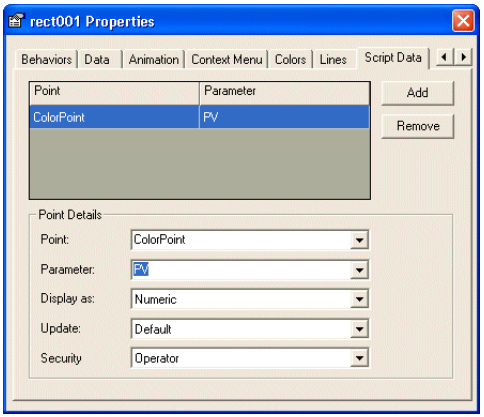


3 Add the objects to required to create the pump.

Don’t forget to group the objects—this is a requirement for a dynamic shape.

4 Select the rectangle whose color you want to control, open the Properties Window, click the Behaviors tab and select **Script Data**.

5 Click the Script Data tab and add the point you defined in step 2 as a custom property.

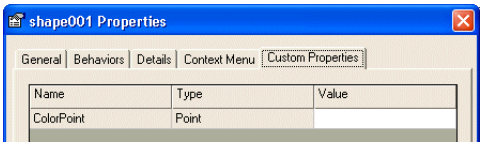


6 With the object still selected, open the Script Editor and, for the ondatachange event, write the following script:

```
Sub rect001_ondatachange
  ' Get the appropriate value
   PumpMode = rect001.GetDataValue "ColorPoint.PV"

  ' Now update the pump's appearance
  if PumpMode = 1 then
    Group.rect001.Color = vbRed
  else
    Group.rect001.Color = vbGreen
  end if
End Sub
```

Inserting the shape into displays

Each time you insert this shape into a display, you select the Custom Properties tab and type the ID of the appropriate point in **Value**.



Using color to reflect the value of a status point

Scenario

You want an object's color to indicate the current value of the PV parameter of a status point.

Solution

You add the point's PV parameter to the object's Script Data tab so that the script can access the parameter. (See "Reading and writing to point parameters" on page 304.)

Scripts

You attach the following script to the object's ondatachange event. In this example, the point is called "poista85", and it has two values: `0` and `1`.

```
Sub rect001_onchange
  If rect001.DataValue("poista85.PV") = 0
    rect001.fillColor=vbRed
Else
    rect001.fillColor=vbGreen
  End if
End sub
```

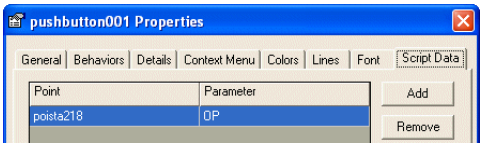
Using a button to start a pump

Scenario

You want to use a button to start a pump. You also want the user to confirm the action before starting the pump.

Solution

You add the parameter that controls the pump to the button's Script Data tab so that the script can access it—in this example, you add the OP of a status point called "poista218". (To learn how to do this, see "Adding point parameters to an object" on page 203.)



You use the GetResponse3 method to display a prompt in Station's Message Zone, and to obtain the user's Yes/No response.

Scripts

You attach this script to the button's onclick event. (This script set up a timer. As specified in the description for the GetResponse3 method, you must only use the method within a timer subroutine.)

```
Sub pushbutton001_onclick
  m_iTimerID = window.setInterval ("TimerHandler()", 1000)
End Sub
```

You add this timer subroutine to the general section.

```
Sub TimerHandler()
  Dim str, iStatus
  If pushbutton001.DataValue("poista218.OP") = 0 then
    iStatus = window.external.GetResponse3 ("Are you sure that
   you want to start this pump ", 1 , str)
    If iStatus = 1 then
      If str = "" then
        'Haven't received a response yet
       ' .
        '.
      Else
        'Action confirmed.
        If str = "Y" then
          pushbutton001.fillColor = vbGreen
```

```
            pushbutton001.DataValue("poista218.OP") = 1
          End If
        End If
      Else
        'Action canceled, so kill timer.
        window.clearInterval(m_iTimerID)
      End If
    End If
        pushbutton001.fillColor = vbRed
End Sub
```

Sending a display to another Station in a console

Scenario

You want an operator to be able to send a particular display to a specific Station within the console by clicking an alphanumeric on the display.

Solution

You add a script to the alphanumeric for the onclick event to request LRN 21.

Scripts

```
Sub alpha001_onclick
  window.external.RequestTask 21, 15, 1, “CSTN02-1”, 0, 0
  window.external.RequestTask 21, 1, “BoilerPageCallup.htm”, 0,
   0, 0
End Sub
```

Remarks

- You could change the script to send the current display as follows:
  `window.external.RequestTask 21, 15, 0, “CSTN02-1”, 0, 0`
- You could use an alias to specify the Station to which the display is spent. For more information about aliases, see the section “Configuring Console Stations and Consoles” in the *Configuration Guide*.

Creating a jogging control

Scenario

You want a “jogging” control for a motorized valve. (A jogging control gives the user accurate control over the valve’s position.)

Solution

You add a button and write scripts for its onmousedown and onmouseup events so that the motor runs while the left mouse button is held down.

You add the SP parameter of the valve’s control point (called “ValveControl”) to the button’s Script Data tab, so that you can write to the parameter. (See “Reading and writing to point parameters” on page 304.)

Scripts

This script is attached to the button’s onmousedown event.

```
pushbutton001.DataValue("ValveControl.SP") = 1
pushbutton001.fillColor = vbGreen
```

This script is attached to the button’s onmouseup event.

```
pushbutton001.DataValue("ValveControl.SP") = 0
pushbutton001.fillColor = vbRed
```

Performing an action in response to a right-click

Scenario

You want to call up the previous display when the user right-clicks somewhere on the display.

Solution

You use the onmousedown event to detect which button was clicked. When the event fires, the value of `window.external.button` is checked (a value of `2` indicates that the right button was clicked).

If this is the case, LRN 21 (the Server Display program) is requested with parameter `21` (the "page back" command).

Scripts

The script is attached to the onmousedown event for the Page.

```
If window.event.button = 2 Then
  window.external.requestserverlrn 21,21,0
End If
```

Remarks

- You cannot use the onclick event.

Checking the user's security level

Scenario

You want to prevent access to certain information if the person does not know the manager-level password.

Solution

You use a button to control the display of the information. The script first checks the current security level by reading the value of `window.external.SecurityLevel`.

If the current level is not manager, the user is prompted to for the manager-level password. This is done via the `window.external.executeoperatorcommand("psw")` command.

Scripts

This script is added to the button's onclick event.

```
Sub cmdDetails_onclick
  '0 = LVL 1
  '1 = LVL 2
  '2 = Operator
  '3 = Supervisor
  '4 = Engineer
  '5 = Manager
  Dim Level
  'Set Level variable on page loading
  Level = window.external.SecurityLevel
  'Check for presence of manager mode.
  If Level = 5 Then
    'display manager information in Message Zone.
    window.external.MessageZoneText = "Manager Logged On"
    'Display manager-only information.
    alpha5.style.visibility = "visible"
    .
    .
  Else
window.external.MessageZoneText = "Insufficient Security
   Clearance. Please log on as a manager."
    window.external.executeoperatorcommand ("psw")
  End if
End Sub
```

Remarks

- Alternative ways of displaying the manager-only information include a message box and a popup.
- This scenario assumes Station-based security. If you have operator-based security and the user does not have manager-level access, an appropriate message might be: "You do not have sufficient security clearance".

Starting an application by clicking a button

Scenario

You want an application to start when the user clicks a button.

Solution

This example starts Microsoft Word using the CreateObject command. (The CreateObject command requires the object's `name` and `type`. In the case of Microsoft Word, these are `Word` and `Application`.)

Having started Word, its object model can be accessed via the variable, `msword`. In this example, its `visible` property is set to true to show the application.

Scripts

This Script is attached to button's onclick event.

```
Sub cmdNotepad_onclick
  Dim msword
  Set msword = CreateObject("Word.Application")
  msword.visible = true
End Sub
```

Controlling Station from another application

Scenario

You want to create a training support tool, written in Visual Basic, that presents trainee operators with a series of interactive flow charts they must work through. To improve usability, the support tool instructs Station to display the appropriate display at various stages in a procedure.

Solution

First, start a new Visual Basic project as a standard exe. Then create two command buttons called, `cmdStart` and `cmdPage80`. The `cmdStart` button starts Station and `cmdPage80` button calls up the required display (page 80).

Scripts

The code for the project's general section.

```
Public objStationApp As Object
Option Explicit
```

The the script for the onclick event of `cmdStart`. Station is started using the CreateObject function. Any parameters that need to be passed to Station are done through the object variable `objStationApp`.

```
Private Sub cmdStart_Click()
  Set objStationApp = CreateObject("Station.Application")
End Sub
```

The script for the onclick event of `cmdPage80`. A command is sent to Station to call up page 80.

```
Private Sub cmdPage80_Click()
  objStationApp.CurrentPage = 80
End Sub
```

Responding to Station events from another application

Scenario

You want to create a visible, separate application that responds to events fired by Station. This application is invoked from a display.

Solution

Write a Visual Basic ActiveX EXE that is invoked within Station via page scripts.

To prepare the ActiveX control:

1 Start a new Visual Basic project as an ActiveX EXE.

2 Reference the Station Type Library.

The ActiveX EXE requires knowledge of the Station Application object, its methods, properties and events. To expose these you need to add the "Station Type Library (Version 2)" to the project references. Station needs to be installed on your computer for its type library to be visible.

3 Add a Station Application variable to your ActiveX EXE.

This variable stores the Station Application object. Declare the variable "withevents" to allow handling of any application events fired by Station.

4 Add a method to allow Station to connect to your ActiveX EXE.

This method is called within script to pass in a pointer to the Station Application object. Your ActiveX EXE stores this point in a global variable for future use.

5 Write Event Handlers for the events you are interested in.

```
Private Sub objStation_OnConnect()
  'Insert code here to handle the OnConnect event
End Sub
```

6 Add scripts to your page to invoke your ActiveX EXE

These scripts create your ActiveX EXE, pass it a reference to the Station Application object, and, if required, instruct the ActiveX EXE to display any forms it contains.

Scripts

The code for the general section of the class module for steps 3, 4 and 6 is shown below for an ActiveX EXE that contains a form called UIForm.

```
Option Explicit
Dim WithEvents m_objStation As Station.Application2
' ************************************************
' This method MUST exist in this module to use Station
   Application
```

```
Public Sub SetStationObject(ByRef objStation As
   Station.Application2)
    Set m_objStation = objStation
    Set UIForm.objStation = objStation
End Sub
' ****************************************************
' ****************************************************
' This method MUST exist in this module to load the form via
   script
Public Sub Load()
    UIForm.Visible = True
End Sub
' ****************************************************
```

Once your ActiveX EXE is compiled you can write your page script that creates and loads it.

```
Set MyControl=CreateObject("MyControl.clsMyControl")
MyControl.SetStationObject(window.external)
MyControl.Load()
```

Forcing a display to use a particular SafeView window

Attention

This example is only applicable if you use Station in conjunction with SafeView.

Scenario

You want to add several buttons to the display, each of which forces a particular trend display to appear in the "TrendWindow" window. (This window is defined in your workspace configuration file.)

Solution

You use the VBScript GetObject function to create a reference to SafeView. (The GetObject function requires SafeView's unique ProgID, which is `Honeywell.Workspace.Client`.)

Having created a reference to SafeView, you can then control it through its object model. (For details, see the *SafeView User's Guide*.)

In this example, you use IsManagerActive to check whether SafeView is running, and then use SetOutputFocus method to set the next window to receive a display. Finally, you call up the display.

Scripts

You attach this script to each button's onclick event. (Each script will specify a different display.)

```
Sub Pushbutton001_onclick
  dim sview
  set sview = GetObject("" ,"Honeywell.Workspace.Client")
  If sview.IsManagerActive = true then
    dim bResult
    bResult = sview.SetOutputFocus("TrendWindow")
  End if
window.external.CurrentPage = "trend001.htm"
End Sub
```

Remarks

- For details about the GetObject function, see the help for VBScript.

Changing a trend's contents at run time

Scenario 1

You have a trend that has the legend hidden and you want operators to add the points and parameters to the trend by clicking a button.

Solution

Add the trend to the display and hide the legend. Add a button to the display and attach a script to the button's onclick event.

Scripts

```
Sub pushbutton001_onclick
  trend001.Legend.PointID(1) = "sinewave"
  trend001.Legend.Parameter(1) = "pv"
End sub
```

Scenario 2

You have several similar processes and you want an operator to be able to call up trend data for a particular processes by clicking a button.

Solution

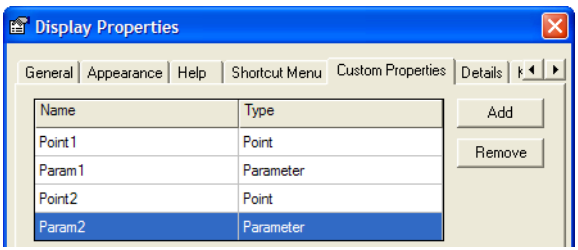
You place the trend in a display and configure the trend so that it is linked to the system's custom properties (See "Accessing the point mapped to a system custom property" on page 312.)

You write scripts for the buttons that specify which point parameters are assigned to the custom properties (and, therefore, to the chart).

To create the display:

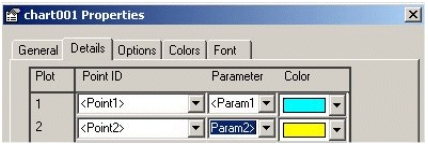
1 Choose **File > New > Display**.

2 Click the Custom Properties tab on the Properties Window and add the following properties. (It is a good idea to give each property an intuitive name.)



3 Add a trend to the display.

4 With the trend selected, click the Details tab on the Properties Window and assign the custom properties to the plots as required.



5 Configure the trend's other properties as appropriate, such as the number of samples, units and so on.

Scripts

Each button's script specifies the appropriate points. This script specifies the points as "POINTANA1" and "POINTANA2".

```
Sub pushbutton001_onclick
  ' Prevent the shape being updated as each custom property is
   changed
  ' Set the custom properties
  displayDataRepository.PutValue "Point", "Point1", "POINTANA1"
  displayDataRepository.PutValue "Parameter", "Param1", "PV"
  displayDataRepository.PutValue "Point", "Point2", "POINTANA2"
  displayDataRepository.PutValue "Parameter", "Param2", "PV"
  ' Apply the changes to the shape
End Sub
```

Remarks

- The chart is reset each time the display is called up or refreshed. (If you don't specify default values for the custom properties, the chart will remain empty until the operator clicks a button.)

Showing and hiding objects

Scenario

You have a display that is difficult to use because it shows too much information. You want to simplify the display by hiding non-critical information unless specifically requested by the user.

Solution

You use a button to toggle the visibility of some objects within a group. (For example, alphanumerics containing non-critical information appear when the button is clicked once and disappear when it is clicked a second time.)

Scripts

The script for the button's onclick event. It controls the visibility of the relevant child objects. (One object, `alpha5`, is used to check whether the objects are visible.)

```
Sub cmdPushbutton_onclick
  If imgGroup1.all("alpha5").style.visibility = "visible" Then
    imgGroup1.all("alpha5").style.visibility = "hidden"
    imgGroup1.all("alpha6").style.visibility = "hidden"
    imgGroup1.all("alpha7").style.visibility = "hidden"
    .
    .
  Else
    imgGroup1.all("alpha5").style.visibility = "visible"
    imgGroup1.all("alpha6").style.visibility = "visible"
    imgGroup1.all("alpha7").style.visibility = "visible"
    .
    .
  End If
End Sub
```

Using event bubbling to reduce script effort

Scenario

You want the mouse pointer to change to a hand symbol over all pushbutton objects.The page that contains a large number of objects, many of which are scripted almost identically and you want to avoid having to write the same script for each pushbutton object.

Solution

Instead of scripting the onmouseover and onmouseout events for all pushbuttons on the page, script these events on the page object. In the event handler, check the ID of the object firing the event and, if it indicates the object is a pushbutton, change the mouse pointer accordingly.

Scripts

The script for the page's onmouseover event.

```
Sub Page_onmouseover
  If (Left(window.event.srcElement.id, 10) = "pushbutton") Then
   window.event.srcElement.style.cursor = "hand"
  End If
End Sub
```

The script for the page's onmouseout event.

```
Sub Page_onmouseout
  If (Left(window.event.srcElement.id, 10) = "pushbutton") Then
    window.event.srcElement.style.cursor = "auto"
  End If
End Sub
```

Remarks

- For a description of style.cursor, see "style object" on page 407.

Index

A

acronyms
  custom 240
  default 240
  described 240
  system 240
  user 240
active object, described 142
ActiveX controls
  inserting 101
  supplied with HMIWeb 338
ActiveX documents, inserting 102
Add method 422
AddNewPlot method 423
AddPlotData method 425
AddString method 427
alarm and event tables
  described 237
  properties
    Data tab 175
    Details tab 179
    Sorting tab 208
alarm and event tables and tables
  properties
    Filters tab 193
aligning
  objects 80
  text 31
all collection 294
Alphanumeric object 339
Animation tab, Properties Window 163
animation, using a shape sequence as 56
Appearance tab, Properties Window 244
Application object 342
Appwindow object 350
archive files
  archiving/unarchiving 271
  described 65
Arrange toolbar 28
arranging objects 28

B

batch editing displays 271
behaviors
  defining 86
  described 86
  tab, Properties Window 165
Bezier curves
  creating 70
  curve control handles 71
  described 70
Blue method 429
blur method 430
breakpoints 122
  performance issues 149
bubbling of events 287
bulk editing displays 271
buttons, toolbar 22
  arranging 28
  drawing 32
  editing 25
  formatting text 31
  object creation 25
  scripting 33
  transforming 29
  viewing 30

C

Callup Task tab, Properties Window 245
CancelChange method 431
CancelEvent method 432
CancelOperatorCommand method 434
CancelResponse method 435
Chart object 372
charts
  Axis View 227
  described 223

invalid data 226
Plot View 224
reference line and reference values 224
samples and sample intervals 224
scripting
examples 646
reference 372
ToolTips, plot 224
zoom function 225
Checkbox object 355
children collection 294
ClearPlotData method 436
clearTimeout method 433, 437
click method 438
collections
all 294
children 294
Dictionary 360
Nodes 378
StationWindows 319
colors
defining 103
gradient 105
non-standard 104
setting in style sheets 118
setting/returning in scripts 303
system defined 164
tab, Properties Window 167
columns
tab, Properties Window 169
combining objects
to create a new vector graphic object 72
to improve display performance 83
Combobox object 357
commands, Station 128
comparing strings 322
components, script 286
Connect method 439
Copy method 440
copying example scripts 647
copying objects 24
CreatePopupWindow method 442
CreateTimer method 443
creating
display 45
displays 44
displays from Web pages 45
dynamic shapes 46
faceplates 50
objects 68, 70, 72
popups 49
shape sequences 48
cursor, behavior of 267
curves, Bezier
control handles
described 71
editing 88
described 70
custom properties
assigning to dynamic objects 46
defining 46
described 57
mapping dynamic objects to points 57
mapping points/parameters when inserting a dynamic shape 97
scripts
accessing mapped points 310
re-mapping points at run time 314, 315
storing user-defined data 317, 318
usages 309
tab, Properties Window 168
Custom Properties tab, Properties Window
display 249
cutting objects from the display 24

D

Data tab, Properties Window
alarm or event table 175
other objects 170
data-bound objects (dynamic objects) 170
debugging
error handlers, using 323
reading and writing to points 308
DeleteString method 445
deleting objects 76
deselecting objects 75
Details tab, Properties Window 179, 191
display 250
object 176
Dictionary collection 360
display scripts, described 285
DisplayDataRepository object 361
DisplayExists method 446
displays
archive format 65
creating 44

- distributing 279, 281
- help, assigning 258
- performance
  - analyzing 138
  - guidelines 141
  - optimizing 137
- point detail 51
- printing 24
- properties 243, 244, 245, 249, 250, 253, 258, 261, 262
- scripts 285
- source, showing/hiding 30
- standard, creating 45
- template (point detail) 51
- templates 60
- types 53
- zooming in and out 30
- zooming in and out of 25

distributing
- displays 279
- objects 81

distributing displays 281
document (help), assigning to a display 260
DPToLPX method 447
DPToLPY method 447
drawing techniques 67
Drawing toolbar 32
DSP displays
- linking to 221
- scripting differences 324
- scripts for 285

duplicating objects 78
dynamic objects
- custom properties 57
- described 170

dynamic shapes
- creating 46
- custom properties
  - assigning to dynamic objects 46
  - defining 46
  - described 57
  - mapping points/parameters 97
- described 54
- inserting 97

E

editing a display's HTML source 30
editing buttons 25, 30
embedding shapes
- described 98
- linked to the display 98
- optimizing performance 144
- updating 271
- upgrading 99

error handlers, using 323
event and alarm tables, described 237
event object
- cancelling default action 606
- described 362
- event bubbling 287
- scripting differences between HMIWeb and DSP displays 324
- used to reference an object 293

events
- bubbling 287
- handlers 287
- onactivate 564
- OnActivated 565
- OnAlarm 566
- OnAppStartup 567
- OnAppStatusUpdate 568
- OnBeforeConnect 569
- OnBeforeOKBKeyPressed 570
- OnBlink 571
- onblur 572
- onchange 573
- OnChartLoadComplete 637
- OnChartScroll 640
- OnChartUnZoomed 638
- OnChartZoom 639
- onclick 574
- OnConnect 575
- oncontextmenu 576
- OnCustomEvent 577
- ondatachange 579
- ondblclick 581
- ondeactivate 582
- OnDictionaryValueChanged 583
- OnDisconnect 584
- OnDisplayNotFound 585
- onfocus 588
- onkeydown 589
- onkeypress 590
- onkeyup 591
- OnMenu 592
- OnMenu2 593
- onmouseenter 594, 595
- onmouseleave 596

onmousemove 597
onmouseout 598
onmouseover 599
onmouseup 600
OnOKBKeyPressed 601
OnOperatorChange 602
OnOperatorCommand 603
OnOperatorCommand2 604
onoperatorkey 606
onpagecomplete 611
OnPageLoad 612
OnPeriodicUpdate 614
OnQuit 617
OnReferenceLineSet 644
OnResponse 622
onshapeload 626
onshapeunload 627
OnSilenceIKB 628
OnTimer 629
onunload 632
onupdate 633
OnUserObjectNotify 634
example scripts 645
ExecuteOperatorCommand method 448

F

faceplates and popups
attaching to an object 100
behaviors 165
creating
faceplate 50
popup 49
described 59
scripts
changing the size 298
changing the size of the content 299
closing the popup 298
CreatePopupWindow method 442
example scenarios 648
FindPopup method 450
referencing the display or Station 294
tips and techniques 298
files
auto recovery 267
distributing 279
properties 243
recently used 267
fill, gradient 105
Filters tab, Properties Window 193
FindFile method 449
finding/replacing in scripts 331
FindPopup method 450
FlagUserInteraction method 452
flipping objects 84
focus method 451
Font tab, Properties Window 194
Format toolbar 31

G

general section (scripts) 290
General tab, Properties Window
display 253
object 195
generic displays, about 61
GetConnectionProperty method 453
GetCustomProperty method 454
GetPlotColor method 465
GetPlotDiscrete method 466
GetPlotMarkerStyle method 467
GetPlotStepMode method 468
GetPlotVisible method 469
GetReferenceValues method 471
GetResponse method 473
GetResponse2 method 475
GetResponse3 method 477
GetResponse4 method 479
GetStyleClassProperty method 483
GetStyleProperty method 481
gradient fill 105
graphics
formats supported 93
inserting 92
Green method 491
grid
setting gridline spacing 268
showing/hiding 30
snapping to 30
Group object 366
groups
adding objects to 82
combining 83
creating 82
referencing child objects 293
referencing child objects with all and children collections 294

referencing objects within 293
removing objects from 82
guidelines, display performance 141

H

handles
Bezier curve control 71
reference object 75
rotation 85
selection 74
help
defining for a display 258
tab, Properties Window 258
help for scripting languages 286
HMIWeb Display Builder Assistant 271
HMIWeb Object Model, overview 334
hover behavior 166, 195
HTML Help, assigning a topic 259
HTML source of a display 30
Hyperlink object 368
hyperlinks
creating 69
specifying URLs 221

I

improving the performance of an existing display 152
Indicator object 370
inserting
ActiveX controls 101
ActiveX documents 102
dynamic shapes 97
graphics 92
shape sequences 96
text 95
Intellisense, described 329
InvokeCommand method 495
InvokeMenu2 method 496
InvokePopup method 497
InvokeShortcutMenu 135
Item method 498

J

JScript 286

K

Keys tab, Properties Window 261
keys, shortcut 277
keywords
me (VBScript) 293
this (JScript) 293
KillTimer method 500

L

languages, scripting 286
layout of Display Builder 21
libraries, shape
registering 269
supplied 41
lines
style 32
tab, Properties Window 197
thickness 32
linking
DSP format displays 221
dynamic shape 97
files and Web sites 221
objects to the database 170
shape sequence 96
locking objects 79
log file, Station 501
LogMessage method 501
LPToDPX method 502
LPToDPY method 502
LRN 262

M

main window 21
MakeColor method 503
mapping dynamic objects to points 57
me keyword (VBScript) 293
menus
HMIWeb Display Builder's shortcut 278
shortcut
creating 126
described 125
invoking through scripts 135
scripts, writing 130
structure of 127
MessageBox method 504

methods
Add 422
AddNewPlot 423
AddPlotData 425
AddString 427
Blue 429
blur 430
CancelOperatorCommand 431, 432, 434
CancelResponse 435
ClearPlotData 436
clearTimeout 433, 437
click 438
Connect 439
Copy 440
CreatePopupWindow 442
CreateTimer 443
DeleteString 445
DisplayExists 446
DPToLPX 447
DPToLPY 447
ExecuteOperatorCommand 448
FindFile 449
FindPopup 450
FlagUserInteraction 452
focus 451
GetConnectionProperty 453
GetCustomProperty 454
GetPlotColor 465
GetPlotDiscrete 466
GetPlotMarkerStyle 467
GetPlotStepMode 468
GetPlotVisible 469
GetReferenceValues 471
GetResponse 473
GetResponse2 475
GetResponse3 477
GetResponse4 479
GetStyleClassProperty 483
GetStyleProperty 481
Green 491
InvokeCommand 495
InvokeMenu2 496
InvokePopup 497
Item 498
KillTimer 500
LogMessage 501
LPToDPX 502
LPToDPY 502
MakeColor 503
MessageBox 504
Objects 505
Quit 507
RaiseCustomEvent 508
Red 509
Redraw 510
Refresh 511
ReloadContent 512
Remove 513
RemovePlot 514
RemovePlotData 515
RequestDemandData 516
RequestServerLRN 518
RequestTask 519
ResetContent 521
SendKeyPress 524
SetCommandUIState 525
SetCustomProperty 526
SetInterval 532
SetNodePoint 536
SetPlotColor 539
SetPlotDiscrete 540
SetPlotMarkerStyle 541
SetPlotPen 542
SetPlotRange 543
SetPlotStepMode 545
SetPlotVisible 546
SetTimeout 549
SetTitleBarFormat 551
Shell 560
UnZoom 561
UserObjectNotify 562
minimizing the object count 142
mirror-imaging objects 84
moving objects 91
multi-window environment, writing scripts for 319

N

Node Editor
described 88
using 88
Node object 377
Nodes object 378
nodes of an object, described 88
nudging objects 91

O

Object Explorer
  described 34
  showing/hiding 30
objects
  active, described 142
  aligning 80
  Alphanumeric 339
  Application 342
  AppWindow 350
  arranging and aligning 28
  behaviors 86
  changing shape 88
  Chart 372
  Checkbox 355
  colors 103, 104, 105
  combining
    to create a new vector graphic object 72
    to improve display performance 83
  Combobox 357
  copying 24
  creating 25, 68, 70, 72
  curve control handles 71
  deleting 24, 76
  deselecting 75
  Dictionary 360
  dimensions, specifying 321
  DisplayDataRepository 361
  distributing 81
  duplicating 78
  dynamic
    custom properties 57
    described 170
  event 362
  faceplate, attaching 100
  flipping 84
  Group 366
  grouping 82
  handles, selection 74
  Hyperlink 368
  Indicator 370
  linking to the database 170
  locking 79
  make same size 77
  mirror-image 84
  moving 91
  Node 377
  Nodes 378
  nodes 88
  nudging 91
  Object Explorer 34
  Page 380
  pasting from the clipboard 24
  Picture 382
  Popup 390
  popup, attaching 100
  position, specifying 321
  properties 38, 161
  Pushbutton 391
  reference object 75
  referencing in scripts 293
  resizing 77
  rotating 85
  RuntimeStatus 394
  selecting 34, 74
  Shape 398
  sizing 77
  stacking order, rearranging 87
  Station 401
  StationDataRepository 404
  StationWindows 405
  style 407
  Text 408
  transforming 29
  uncombining 83
  ungrouping 82
  Vector Graphic 413
  Windows 416
Objects method 505
onactivate event 564
OnActivated event 565
OnAlarm event 566
OnAppStartup event 567
OnAppStatusUpdate event 568
OnBeforeConnect event 569
OnBeforeOKBKeyPressed event 570
OnBlink event 571
onblur event 572
onchange event 573
OnChartLoadComplete event 637
OnChartScroll event 640
OnChartUnZoomed event 638
OnChartZoom event 639
onclick event 574
OnConnect event 575
oncontexmenu event 576
OnCustomEvent event 577

ondatachange event 579
ondblclick event 581
ondeactivate event 582
OnDictionaryValueChanged event 583
OnDisconnect event 584
OnDisplayNotFound event 585
onfocus event 588
onkeydown event 589
onkeypress event 590
onkeyup event 591
OnMenu event 592
OnMenu2 event 593
onmouseenter event 594, 595
onmouseleave event 596
onmousemove event 597
onmouseout event 598
onmouseover event 599
onmouseup event 600
OnOKBKeyPressed event 601
OnOperatorChange event 602
OnOperatorCommand event 603
OnOperatorCommand2 event 604
onoperatorkey event 606
onpagecomplete event 611
OnPageLoad event 612
OnPeriodicUpdate event 614
OnQuit event 617
OnReferenceLineSet event 644
OnResponse event 622
onshapeload event 626
onshapeunload event 627
OnSilenceIKB event 628
OnTimer event 629
onunload event 632
onupdate event 633
OnUserObjectNotify event 634
options, HMIWeb Display Builder 266

P

Page object 380
page-level scripts that access Station-related properties 300
pasting
  graphic 92
  objects from the clipboard 24
  text 95
performance, display
  ActiveX controls 150
  analyzing 138
  guidelines 141
  improving an existing display 152
  optimizing 137
  sizing of display 150
  Station-related guidelines 151
periodic tasks
  described 262
  tab, Properties Window 262
Picture object 382
pictures
  inserting 92
pictures, inserting 92
Point Browser
  described 36
  showing/hiding 30
point detail displays 51
points
  debugging scripts that access points 308
  linking to objects 171
  Points Browser 36
  reading/writing with scripts 304
  re-mapping at runtime 314, 315
  renaming 271, 274
Popup object 390
Popup tab, Properties Window 199
popups and faceplates
  attaching to an object 100
  behaviors 165
  creating
    faceplate 50
    popup 49
  described 59
  scripts
    changing the size 298
    changing the size of the content 299
    closing the popup 298
    CreatePopupWindow method 442
    example scenarios 648
    FindPopup method 450
    referencing the display or Station 294
    tips and techniques 298
preview of a display
  as it appears in Station 24
  as it appears when printed 24
print preview 24

printing 24
productivity tools 270, 271
properties
  described 38
  display 243
  object 161
Properties Window
  described 38
  display properties 243
  object properties 161
  showing/hiding 30
  tabs
    Animation 163
    Appearance 244
    Behaviors 165
    Callup Task 245
    Colors 167
    Custom Properties (display) 249
    Custom Properties (object) 168
    Data 170
    Data (alarm or event table) 175
    Details (display) 250
    Details (object) 177, 178, 179, 181, 183, 184, 185, 186, 187, 188, 190, 191
    Filters 193
    Font 194
    General (display) 253
    General (object) 195
    Help 258
    Keys 261
    Lines 197
    Periodic Task 262
    Popup tab 199
    Repeat (object) 200
    Script Data 201
    Scrollbar (object) 204
    Shortcut Menu (display) 263
    Shortcut Menu (object) 207
    Sorting 208
Pushbutton object 391

Q

Quit method 507

R

RaiseCustomEvent method 508
reading point parameters in scripts 304
recovery, auto 267
Red method 509
Redraw method 510
reference object, described 75
referencing objects
  child objects 293
  child objects with all or children collections 294
  display from a popup 294, 298
  parent object 294
  parent object from a popup 298
  Station application from a popup 294, 298
  within a display 293
  within groups or shapes 293
Refresh method 511
registering shape folders 269
ReloadContent method 512
re-mapping custom properties at run time 314, 315
Remove method 513
RemovePlot method 514
RemovePlotData method 515
renaming points 271
Repeat tab (table), Properties Window object 200
replacing/finding in scripts 331
RequestDemandData method 516
RequestServerLRN method 518
RequestTask method 519
ResetContent method 521
resizing objects 77
rotating objects 85
rulers, showing/hiding 30
RuntimeStatus object 394

S

SafeView
  controlling the workspace from a display 670
  window category, defining 252
  window category, selecting 250
scenarios (examples) 645
Script Data tab, Properties Window 201
Script Editor
  default scripting language, setting 286
  described 327
  Intellisense 329
  showing/hiding 30

scripts
components 286
copying examples 647
custom properties, using 309
differences between DSP scripts 324
duplicating 78
examples 645
finding/replacing 331
general section 290
indenting/outdenting 33
multi-window Station 319
page-level that access Station-related properties 300
performance issues 147
reading/writing point parameters 304
scenarios 645
Script Editor 327
scripting language 286
shapes 296
Station-level 320
toolbar 33
writing 330
Scrollbar tab, Properties Window
object 204
searching/replacing in scripts 331
selecting objects 74
selection handles 74
SendKeyPress method 524
SetCommandUIState method 525
SetCustomProperty method 526
SetInterval method 532
SetNodePoint method 536
SetPlotColor method 539
SetPlotDiscrete method 540
SetPlotMarkerStyle method 541
SetPlotPen method 542
SetPlotRange method 543
SetPlotStepMode method 545
SetPlotVisible method 546
SetTimeout method 549
SetTitleBarFormat method 551
Shape Gallery, described 40
Shape object 398
shape sequences
creating 48
described 55
inserting 96
shapes
dynamic, described 54
embedding linked 98
folders, registering 269
inserting 96, 97
libraries, supplied 41
performance issues 144
performance of 145
properties 243
referencing child objects 293
referencing child objects with all and children collections 294
scripts 296
sequence, described 55
Shape Gallery 40
upgrading embedded 99
viewing 40
Shell method 560
shortcut keys 277
Shortcut Menu tab, Properties Window
display 263
object 207
shortcut menus
creating 126
described 125
HMIWeb Display Builder's 278
invoking through scripts 135
scripts, writing 130
structure of 127
sizing objects 77
snap-to-grid 30
Sorting tab, Properties Window 208
SSOs (Station Scripting Objects) 320
stacking order
described 87
rearranging 87
sorting objects by 34
Standard toolbar 24
starting Display Builder 20
Station
help for a display 258
log, adding entries to 501
preview of a display 24
Station commands, finding the names of 128
Station object 401
Station Scripting Objects (SSOs) 320
StationDataRepository object 404
Station-level scripts 320
StationWindows collection 319

StationWindows object 405
status indicator, using a shape sequence as 55
strings, comparing 322
style object 407
style sheets
  creating 111
  described 108
  shape files 113
  structure of a style 112
  style attributes
    definition syntax 112
    predefined 114
    scripting equivalents to predefined 119
    user-defined 121

T

table
  properties
    Details tab 191
tables
  creating 241
  described 241
  modifying 242
Tables Properties Window
  tabs
    Columns (object) 169
tables, creating 241
tables, described 241
tables, modifying 242
task request, specifying 262
template displays (point detail) 51
templates, described 60
text
  aligning 31
  formatting 31
  inserting from another application 95
Text object 408
this keyword (JScript) 293
timers, described 291
toolbars
  Arrange 28
  creating 276
  described 22
  displaying/hiding 23
  Drawing 32
  Format 31
  Scripting 33
  Standard 24
  Toolbox 25
  Transform 29
  View 30
Toolbox toolbar 25
tools, productivity 270
Tooltip (hover behavior) 166, 195
topic, assigning a HTML Help topic 259
Transform toolbar 29
transforming objects 29
types of display 53

U

uncombining an object 83
undoing/redoing changes 106
ungrouping objects 82
UnZoom method 561
upgrading embedded shapes 99
URL formats 221
user file, linking an object to 173
UserObjectNotify method 562

V

VBScript 286
vector graphic objects
  combining 83
  creating from overlapping objects 72
  described 222
  nodes, editing 88
  scripting object 413
View toolbar 30

W

Web pages, creating displays from 45
Window object 416
windows
  main 21
  Properties 38
  Shape Gallery 40
writing scripts 330
writing to point parameters in scripts 304

Z

zooming in and out 25, 30

Honeywell Process Solutions
2500 West Union Hills Drive
Phoenix AZ 85027
USA
www.honeywell.com

EP-DSX164
06/06
© 2006 Honeywell International Inc

Honeywell