Honeywell

Experion PKS

HMIWeb Display Building Guide

EPDOC-XX54-en-431A
February 2015

Release 431

Document	Release	Issue	Date
EPDOC-XX54-en-431A	431	0	February 2015

Disclaimer

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 International Sàrl.

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.

Contents

About this guide

This guide describes how to create custom web-based displays using HMIWeb Display Builder.

You can make it much easier for operators to interpret and control system activity if you create suitable custom displays. With custom displays, you can insert your own graphics (for example, photographs and layout diagrams).

Revision history

Revision	Date	Description
A	February 2015	Initial release of document.

Getting started with HMIWeb Display Builder

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*.

The following topics describe the HMIWeb Display Builder application.

Related topics

“Starting HMIWeb Display Builder” on page 18
“Layout of HMIWeb Display Builder” on page 19
“Toolbars” on page 20
“Object Explorer” on page 29
“ToolBox” on page 31
“Properties grid (HMIWeb displays) ” on page 32
“Properties window (HMIWeb displays)” on page 34
“Point Browser” on page 35
“Shape Gallery” on page 37
“About migrating displays” on page 39

Starting HMIWeb Display Builder

To start HMIWeb Display Builder on the server

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

To start HMIWeb Display Builder on a client computer

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

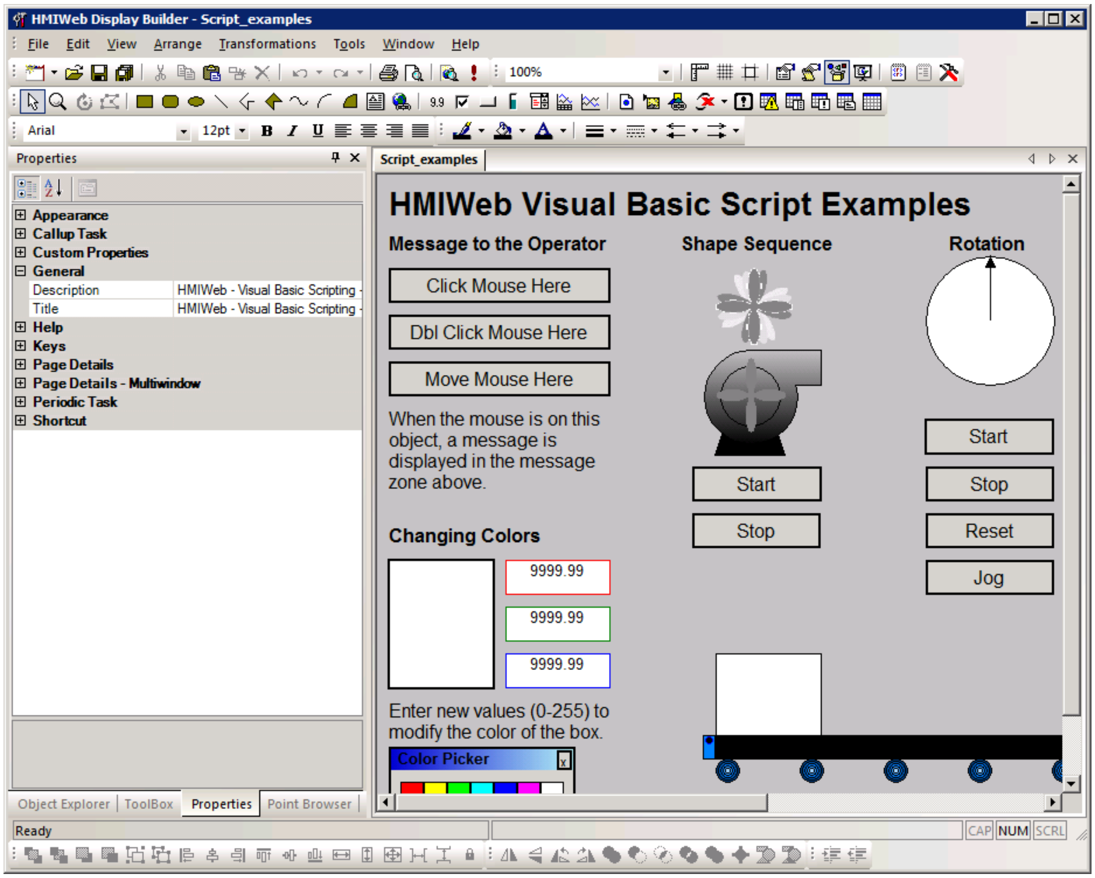
To start HMIWeb Display Builder on a Console Station

- From the **Start** menu, choose **Start** > **All Programs** > **Honeywell Experion PKS** > **Console Station** > **HMIWeb Display Builder**.

Layout of HMIWeb Display Builder

The following figure shows a typical layout, in which:

- Displays you are editing appear on the right. There is a tab for each display.
- The toolbars appear at the top and bottom of the window.
- The tabs for **Object Explorer**, **ToolBox**, **Properties**, and **Point Browser** are displayed, with the focus on the **Properties** tab.



Description: HMIWeb - Visual Basic Scripting -

Title: HMIWeb - Visual Basic Scripting -

When the mouse is on this object, a message is displayed in the message zone above.

Enter new values (0-255) to modify the color of the box.

Figure 1: Typical layout

Related topics

"Object Explorer" on page 29

"ToolBox" on page 31

"Properties window (HMIWeb displays)" on page 34

"Point Browser" on page 35

Toolbars

The toolbars provide speedy access to many functions.

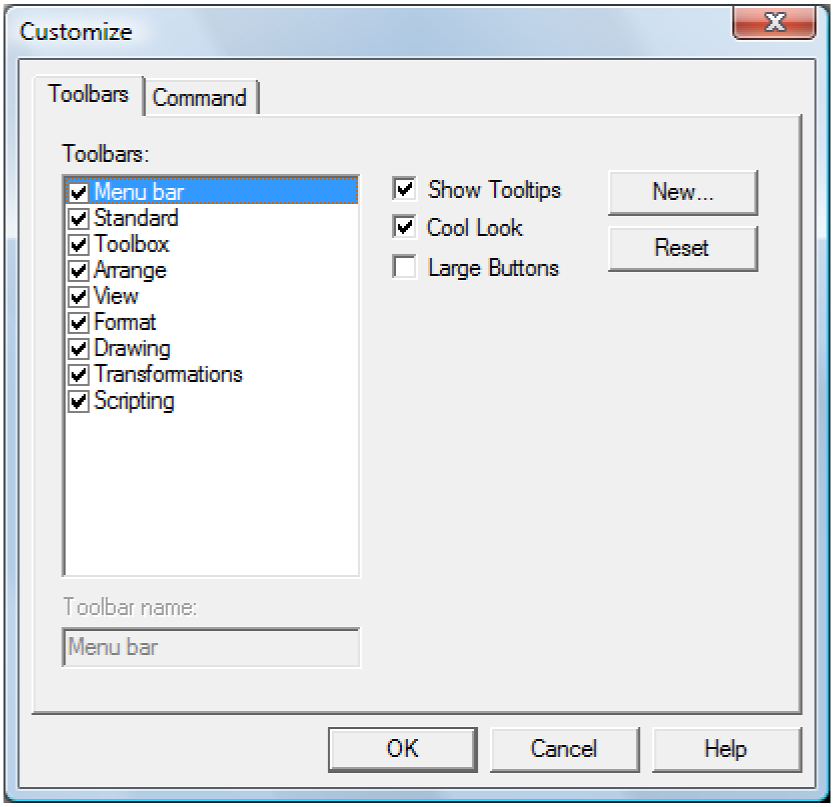
Related topics

“Displaying or hiding toolbars” on page 20
“Standard toolbar” on page 21
“Toolbox toolbar” on page 22
“Arrange toolbar” on page 24
“Transformations toolbar” on page 25
“View toolbar” on page 26
“Format toolbar” on page 27
“Drawing toolbar” on page 27
“Scripting toolbar” on page 28

Displaying or hiding toolbars

To display (or hide) a toolbar

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



Customize

Toolbars:
☑ Menu bar
☑ Standard
☑ Toolbox
☑ Arrange
☑ View
☑ Format
☑ Drawing
☑ Transformations
☑ Scripting

☑ Show Tooltips
☑ Cool Look
□ Large Buttons

Toolbar name: Menu bar

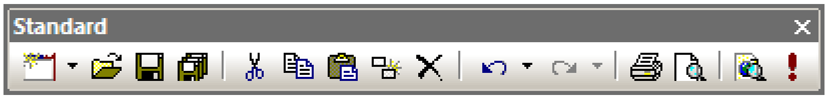
A check mark opposite a toolbar indicates that it is already visible.

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**.



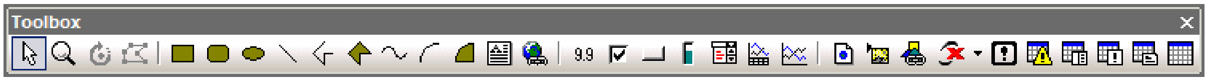
Button	Description
🗋	**New Display**. Creates a new display. Click the arrow to the right of the button to display the list of display types.
📂	**Open**. Opens an existing display.
💾	**Save**. Saves the current display.
💾	**Save All**. Saves all open displays.
✂	**Cut**. Removes the selected object(s) 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.
🖶	**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**.

Button	Description
🔍	**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.
!	**View in Station**. Shows the display in Station with live data supplied by a predefined server. Any scripts in the display will execute in Station.

Toolbox toolbar

Toolbox toolbar buttons

The Toolbox toolbar contains buttons for creating objects on a custom display.



Button	Description
↖	**Pointer**. The main tool you use to select, move and resize objects.
🔍	**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).
⌑	**Node Editor**. Changes the shape of an object by moving or adding individual *nodes* (apexes or reference points).
■	**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.
⬉	**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.

Button	Description
∿	**Bezier Curve**. Creates a smooth-curved line.
◜	**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.
9.9	**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 an automobile's fuel gauge).
▤	**Combo box**. Creates a combo box so that users can select from a list of options in an interactive display.
📈	**Trend**. Creates a trend that displays process values over time in a graphical manner.
📈	**Basic trend**. Creates a simple trend, which has fewer operator controls than a trend.
📄	**ActiveX Document**. Inserts a link to an ActiveX document, such as a Word document, into the display.
🖼	**Picture**. Inserts a picture (graphic).
	Shapelink. Inserts a shape sequence or dynamic shape.
✗	**ActiveX Control**. Inserts a link to an ActiveX Control.
!	**Alarm State**. Inserts an alarm state icon (as used in the Alarm Summary) which shows an alarm state.
⚠	**Alarm Table**. Creates a specialized table that lists alarms.

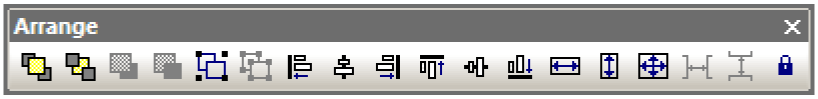
Button	Description
▦	**Event Table**. Creates a specialized table that lists events.
▦	**Alert Table**. Creates a specialized table that lists alerts.
▦	**Message Table**. Creates a specialized table that lists messages.
▦	**Activity Table**. Creates a specialized table that lists activities.
▦	**Table**. Inserts a table into a display.

Related topics

"ToolBox" on page 31

Arrange toolbar

The Arrange toolbar contains buttons for aligning and grouping objects.

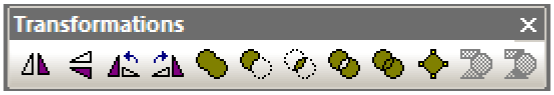


Button	Description
⧉ ⧉ ⧉ ⧉	These buttons change the order in which objects are *stacked* on top of each other. They are: • Bring to Front • Send to Back • Bring Forward • Send Backward
▣ ▣	These buttons group/ungroup objects. They are: • Group • Ungroup

Button	Description
	These buttons align objects. They are: • Align Left • Align Center • Align Right • Align Top • Align Middle • Align Bottom
	These buttons change the size of objects so that they are the same size as a reference object. They are: • Make Same Width • Make Same Height • Make Same Width and Height
	These buttons move objects so that they are evenly spaced. They are: • Even Horizontal Spacing • Even Vertical Spacing
	Locks/unlocks an object. Locking an object protects it from being accidently moved or resized.

Transformations toolbar

The Transformations toolbar contains buttons for transforming objects.

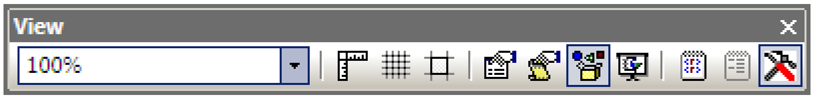


Button	Description
	Flip Horizontal, **Flip Vertical**. These buttons perform mirror-image transformations on objects.
	Rotate Left, **Rotate Right**. These buttons rotate objects +/-90 degrees.

Button	Description
	Union, **Difference**, **Intersection**, **Exclusive OR**, **Join**. These buttons create a new object from two or more overlapping objects.
	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.
	Combine, **Uncombine**. These buttons combine/uncombine static objects, such as lines and rectangles, which helps improve display performance.

View toolbar

The View toolbar includes a range of editing buttons.

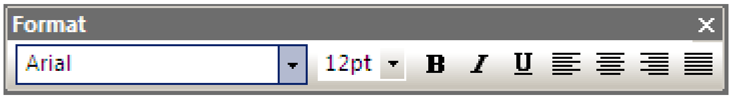


Button	Description
100%	**Zoom**. Sets the magnification for the display. Use the **Zoomer**—🔍—to zoom in on a particular part of the display.
	Rulers. Shows/hides the rulers. The rulers make it easier to arrange objects in a neat and consistent manner.
	Grid. Shows/hides the grid. The grid makes it easier to arrange objects in a neat and consistent manner.
	Snap to Grid. Turns the 'snap to grid' function on and 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, which you use to edit the properties of the selected object(s).
	Script Window. Opens/closes the Script Editor, which you use to write *scripts* (a script is a mini-program that performs a specific task).
	Object Explorer. Shows/hides the Object Explorer, which lists every object in the display.

Button	Description
	Point Browser. Shows/hides the Point Browser, which lists points on a selected Experion server.
	Shape Gallery. Shows/hides the Shape Gallery, which makes it easy to preview and add shapes to your displays.
	HTML Source. Opens the HTML source for the display in a text editor. We strongly recommend that you do not change the source using a text editor because you may corrupt the display.
	Style Sheet. If a style sheet is attached to the display, opens it in a text editor.
	Toolbox. Shows the Toolbox tab, from where you can select objects to draw on the page.

Format toolbar

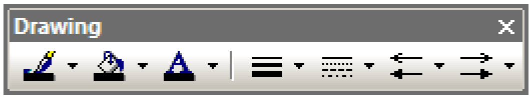
The Format toolbar contains buttons for controlling the appearance of text.



Button	Description
Arial 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	**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

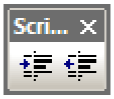
The Drawing toolbar contains buttons for controlling colors and line properties.



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.
▰	**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.
A	**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.
≡	**Line Width**. Sets the line thickness.
┄	**Line Style**. Sets the line style, such as solid, dotted or dashed.
⇇ ⇉	**Start Arrow Style** and **End Arrow Style**. Sets the line arrow style.

Scripting toolbar

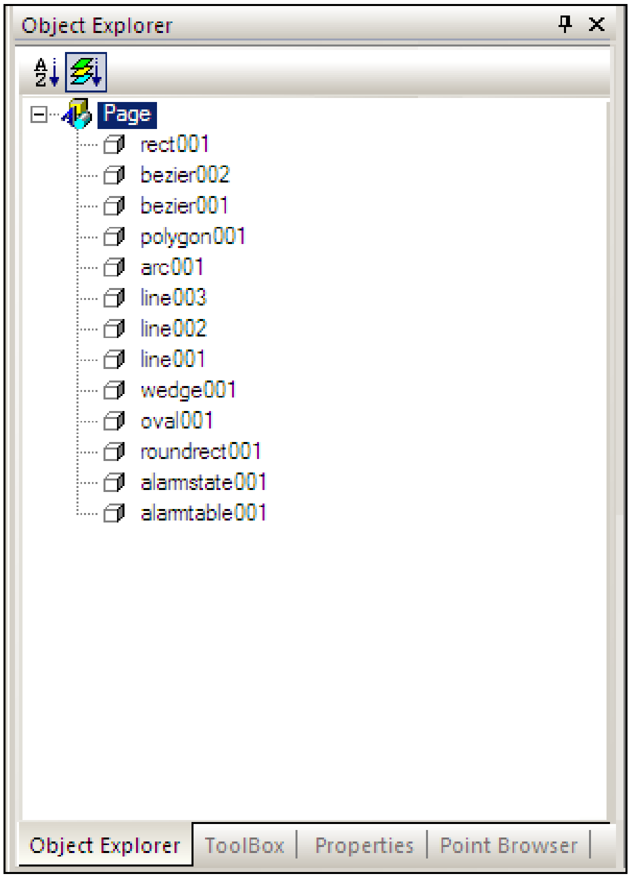
The Scripting toolbar is used in conjunction with the Script Editor to write scripts.



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.



- Page
  - rect001
  - bezier002
  - bezier001
  - polygon001
  - arc001
  - line003
  - line002
  - line001
  - wedge001
  - oval001
  - roundrect001
  - alarmstate001
  - alarmtable001

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

Related topics

"Layout of HMIWeb Display Builder" on page 19

Summary of Object Explorer tasks

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 tab.
Sort objects in alphabetical order	Click AZ↓ on the Object Explorer.

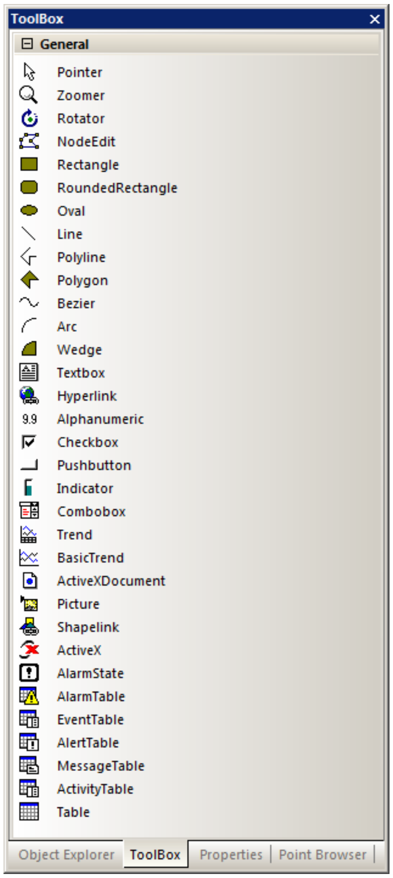
To	Do this
Sort objects in their *stacking* order (the order in which objects are stacked on top of each other)	Click on the Object Explorer. By default, the stacking order is the same as the order in which you created the objects—that is, the first object you created is at the bottom of the stack and the last object is on the top. (You can manually change the stacking order.)
Show/hide objects within a group	Click the '+' or '-' to the left of the group icon.
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.)

ToolBox

The ToolBox contains objects you can use to create custom displays.

Objects that appear in the **General** group on this tab also appear on the Toolbox toolbar. For a description of each object, see the “Toolbox toolbar” topic.

Objects that appear in the **Procedure and Sequence Toolkit** group on this tab are described in the *Procedure and Sequence Custom Display Building Guide*.



ToolBox

⊟ General

Pointer
Zoomer
Rotator
NodeEdit
Rectangle
RoundedRectangle
Oval
Line
Polyline
Polygon
Bezier
Arc
Wedge
Textbox
Hyperlink
Alphanumeric
Checkbox
Pushbutton
Indicator
Combobox
Trend
BasicTrend
ActiveXDocument
Picture
Shapelink
ActiveX
AlarmState
AlarmTable
EventTable
AlertTable
MessageTable
ActivityTable
Table

Object Explorer | ToolBox | Properties | Point Browser

Related topics

“Layout of HMIWeb Display Builder” on page 19

“Toolbox toolbar” on page 22

Properties grid (HMIWeb displays)

Attention

You can change the properties of an object using either the Properties grid or the Properties window. However, due to space considerations, the name of a property on the Properties grid may differ slightly from the name of the same property on the Properties window. In addition, the categorization of properties may also differ. For example, for an alphanumeric object, the **Display as** property appears on the Properties grid in the **Presentation** category, but on the Properties window it appears on the **Details** tab.

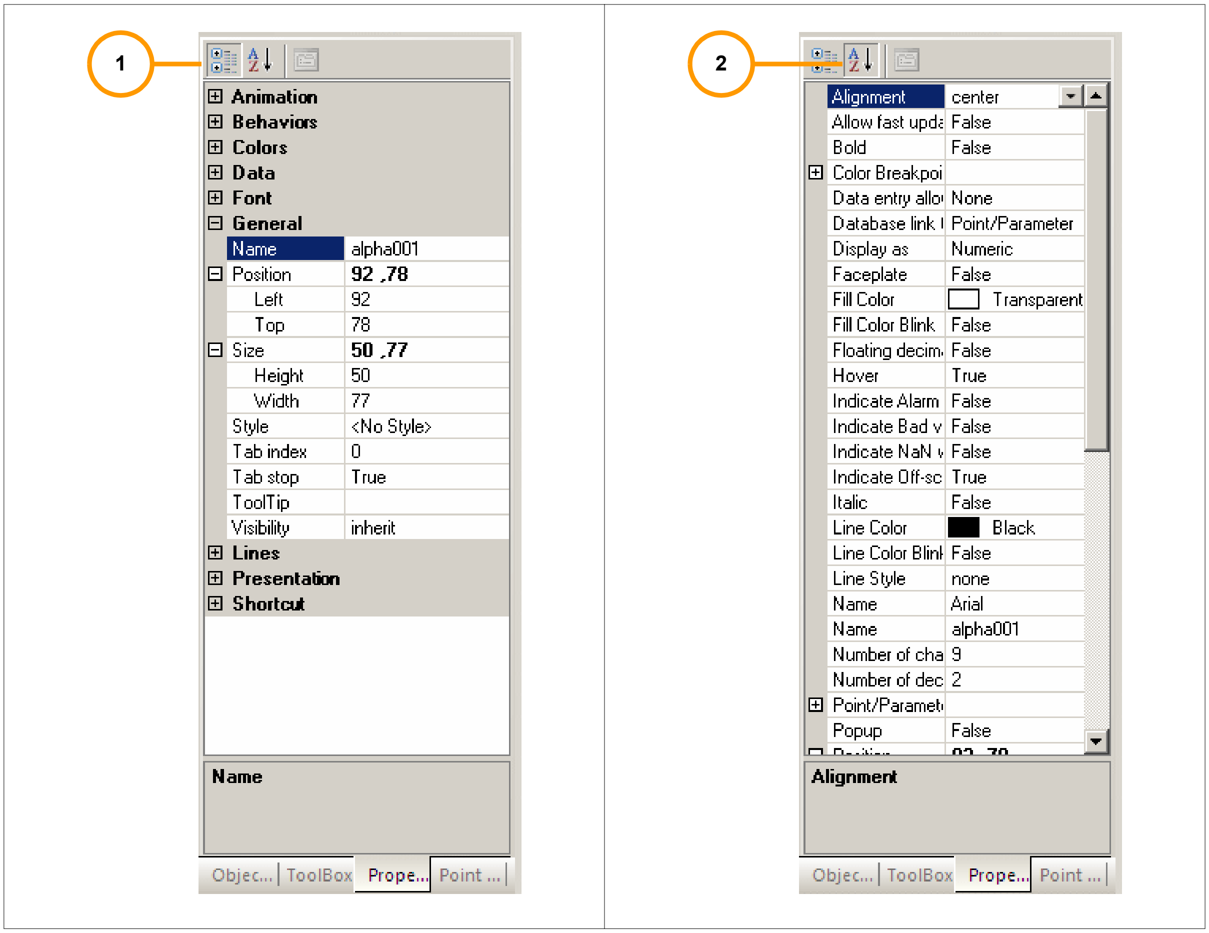
This guide uses the names of properties and categories as they appear on the Properties grid.

You use the Properties grid to view and edit the *properties* of:

- The selected display objects—the properties include physical properties, such as color and size, as well as database link details. The Properties grid changes whenever you select another object.
- The display itself—these appear when no display objects are selected.

The Properties grid can sort properties in two ways: by category or alphabetically.

In the following figure, the example on the left shows the properties for an Alphanumeric display object, sorted by category. The example on the right shows the properties for the same Alphanumeric display object, sorted alphabetically.



⊞ Animation	
⊞ Behaviors	
⊞ Colors	
⊞ Data	
⊞ Font	
⊟ General	
Name	alpha001
⊟ Position	92 ,78
Left	92
Top	78
⊟ Size	50 ,77
Height	50
Width	77
Style	<No Style>
Tab index	0
Tab stop	True
ToolTip	
Visibility	inherit
⊞ Lines	
⊞ Presentation	
⊞ Shortcut	

Alignment	center
Allow fast upda	False
Bold	False
⊞ Color Breakpoi	
Data entry allo	None
Database link I	Point/Parameter
Display as	Numeric
Faceplate	False
Fill Color	□ Transparent
Fill Color Blink	False
Floating decim	False
Hover	True
Indicate Alarm	False
Indicate Bad v	False
Indicate NaN v	False
Indicate Off-sc	True
Italic	False
Line Color	■ Black
Line Color Blink	False
Line Style	none
Name	Arial
Name	alpha001
Number of cha	9
Number of dec	2
⊞ Point/Paramet	
Popup	False

Figure 2: Properties are displayed in categorized or alphabetical view

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

Related topics

“Properties window (HMIWeb displays)” on page 34

Properties window (HMIWeb displays)

Attention

You can change the properties of an object using either the Properties grid or the Properties window. However, due to space considerations, the name of a property on the Properties grid may differ slightly from the name of the same property on the Properties window. In addition, the categorization of properties may also differ. For example, for an alphanumeric object, the **Display as** property appears on the Properties grid in the **Presentation** category, but on the Properties window it appears on the **Details** tab.

This guide uses the names of properties and categories as they appear on the Properties grid.

You can 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.

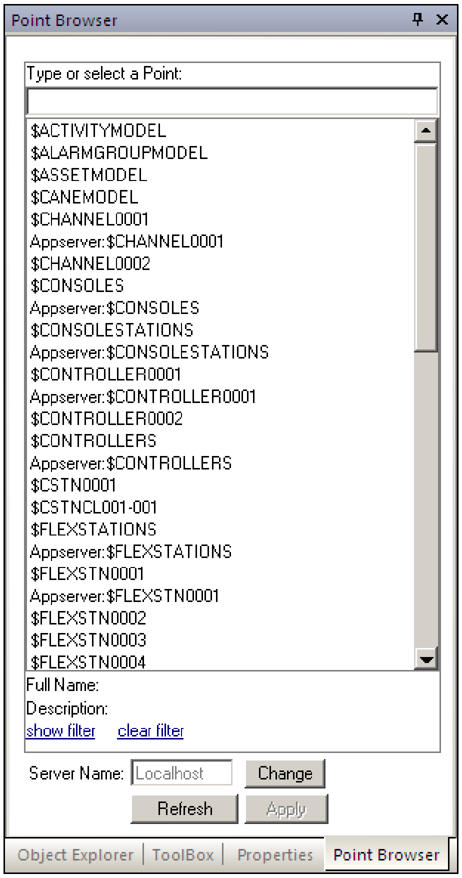
Related topics

"Layout of HMIWeb Display Builder" on page 19

"Properties grid (HMIWeb displays) " on page 32

Point Browser

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



$ACTIVITYMODEL
$ALARMGROUPMODEL
$ASSETMODEL
$CANEMODEL
$CHANNEL0001
Appserver:$CHANNEL0001
$CHANNEL0002
$CONSOLES
Appserver:$CONSOLES
$CONSOLESTATIONS
Appserver:$CONSOLESTATIONS
$CONTROLLER0001
Appserver:$CONTROLLER0001
$CONTROLLER0002
$CONTROLLERS
Appserver:$CONTROLLERS
$CSTN0001
$CSTNCL001-001
$FLEXSTATIONS
Appserver:$FLEXSTATIONS
$FLEXSTN0001
Appserver:$FLEXSTN0001
$FLEXSTN0002
$FLEXSTN0003
$FLEXSTN0004

Server Name: Localhost

Related topics

"Layout of HMIWeb Display Builder" on page 19

Summary of Point Browser tasks

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: • Drag the point name from the point browser onto the 'point' field of the data property page of the display element. • Select the display element then select the point in the point browser and click **Apply**.

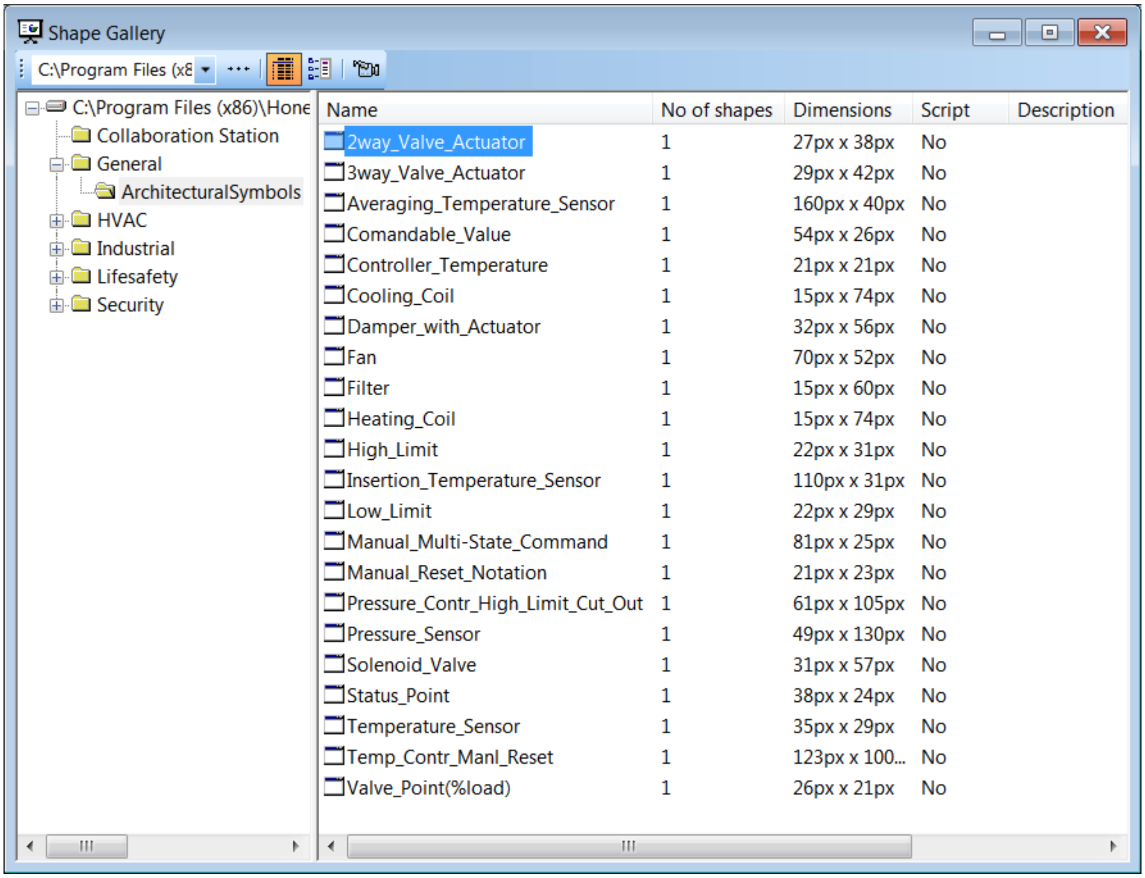
Shape Gallery

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.

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.



Name	No of shapes	Dimensions	Script	Description
2way_Valve_Actuator	1	27px x 38px	No	
3way_Valve_Actuator	1	29px x 42px	No	
Averaging_Temperature_Sensor	1	160px x 40px	No	
Comandable_Value	1	54px x 26px	No	
Controller_Temperature	1	21px x 21px	No	
Cooling_Coil	1	15px x 74px	No	
Damper_with_Actuator	1	32px x 56px	No	
Fan	1	70px x 52px	No	
Filter	1	15px x 60px	No	
Heating_Coil	1	15px x 74px	No	
High_Limit	1	22px x 31px	No	
Insertion_Temperature_Sensor	1	110px x 31px	No	
Low_Limit	1	22px x 29px	No	
Manual_Multi-State_Command	1	81px x 25px	No	
Manual_Reset_Notation	1	21px x 23px	No	
Pressure_Contr_High_Limit_Cut_Out	1	61px x 105px	No	
Pressure_Sensor	1	49px x 130px	No	
Solenoid_Valve	1	31px x 57px	No	
Status_Point	1	38px x 24px	No	
Temperature_Sensor	1	35px x 29px	No	
Temp_Contr_Manl_Reset	1	123px x 100...	No	
Valve_Point(%load)	1	26px x 21px	No	

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

- Do one of the following:
  - Drag it from the list and drop it into the display or
  - Right-click it and choose **Insert into Display** from the shortcut menu.

> **Attention**
> You cannot insert a linked shape unless you have registered the folder that contains the shape.

Related topics

"Registering a shape folder" on page 273

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.)

HMIWeb Solution Pack

The HMIWeb Solution Pack is a comprehensive advanced shapes library that can be used to implement custom displays that are consistent with the Abnormal Situation Management (ASM) Consortium's display guidelines as well as site-specific requirements.

There are two HMIWeb solution packs:

- The *Standard HMIWeb Solution Pack*.
- The *Advanced HMIWeb Solution Pack*, which is licensed, and includes powerful advanced shapes, tabs, and yoking.

The Standard HMIWeb Solution Pack is included on the HMIWeb Solution Pack Installation media, which is included with the Experion media set. For more information, and instructions on installing the Standard HMIWeb Solution Pack, see the *HMIWeb Solution Pack Installation Guide*, which is available on the HMIWeb Solution Pack Installation media.

About migrating displays

After you migrate a client to the new release and before you call up custom or user-defined HMIWeb displays (*.htm* files) in Station, you must use either HMIWeb Display Builder or the Bulk Display Migration tool to update (migrate) the display files.

When you open a display file, (created on an older release) using a new release of HMIWeb Display Builder, the file is updated to the new display format automatically, and then you can save it in the new format.

To migrate multiple HMIWeb displays in bulk rather than individually, you can use the Bulk Display Migration tool (*bulkdisplaymigrator.exe*).

To learn about	See
Migrating HMIWeb display files and HMIWeb faceplates to R431	The topic titled "Display Builder and HMIWeb custom displays" in the *Experion PKS Migration Planning Guide*.
Using the Bulk Display Migration tool	The topic titled "Updating display files" in an Experion PKS migration guide.

About display types

This section lists and describes the various types of display.

Related topics

"Determining the optimum size of a display" on page 42

Creating a display at its optimum size will improve its appearance in different environments.

"About display templates" on page 43

Display templates help you create displays, also ensuring their visual consistency.

"About dynamic shapes" on page 44

A custom object that can be used to present complex dynamic data.

"About shape sequences" on page 45

A custom object that can be used as a status indicator or an animation.

"About custom properties" on page 47

Describes how you can reuse dynamic shapes, shape sequences, and generic displays.

"About popups and faceplates" on page 49

Describes popups and faceplates (a specialized type of popup) and their use in Station.

"About generic displays" on page 51

Explains how generic displays can be reused across your plant.

"About Pan and Zoom displays" on page 54

"About Collaboration Station Workspace displays" on page 55

"About Equipment displays and templates" on page 57

"About archive format" on page 58

A way to save a display or shape in a single file.

"About display definition files" on page 59

Details of the files generated automatically for each display.

"Associating a custom display with an alarm group" on page 60

If you associate a custom display with an alarm group, an icon will appear on the tab for that display when its group contains an active alarm.

Determining the optimum size of a display

Creating a display at its optimum size will improve its appearance in different environments.

The appearance of a display can vary slightly depending a number of factors, such as Station's screen resolution, whether Station uses SafeView or whether the display is called up in a browser.

For example, a display that renders 'perfectly'—no scroll bars and no need to zoom—when called up on one Station will have scroll bars if it is called up on another Station with a lower screen resolution.

Although operators can use the 'zoom to fit' function, this may result in a blank section, either to the right of or below the display. (This effect is caused by the fact that each screen resolution has a different aspect ratio; whereas, the zoom to fit function retains the display's aspect ratio.)

Consequently, if the displays you want to create are likely to be affected by such factors, you should first determine the optimum display size by, for example, creating a simple display and testing it on a range of Stations. When you have determined the optimum size, you should then create a template display that is set to that size.

About display templates

Display templates help you create displays, also ensuring their visual consistency.

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. The default templates folder is located in

`<data folder>\Honeywell\Experion PKS\Client\HMIWeb Display Builder\Templates`, where `<data folder>` is the location where Experion data is stored. For default installations, `<data folder>` is `C:\ProgramData`.

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.

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.

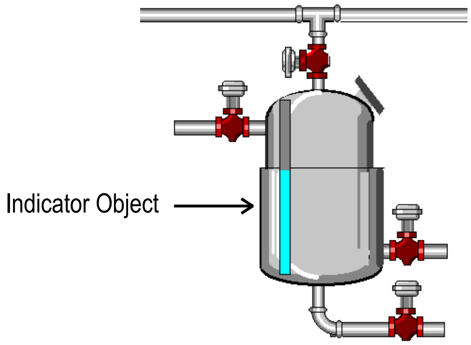
Template	Description
Faceplate display	A faceplate is used to display point information for all points of a particular type. You only need to create a faceplate if you have specialized point types. (Experion is supplied with faceplates for common point types.)
Normal display	A normal display.
Pan and Zoom display	A Pan and Zoom display is several times larger than a normal display and is typically too large to fit on a monitor. Navigation aids enable the Operator to move and zoom into segments of the display to see detailed information. An example of a display that would benefit from this functionality is a map of a long pipeline with equipment at various locations.
Point Detail	A point detail display is used to display point information for all points of a particular type. You only need to create point detail displays if you have any specialized point types. (Experion is supplied with point detail displays for common point types.)
Collaboration Station Workspace display	A Collaboration Station workspace is used to gather a range of key content so that it is easily accessible during a collaboration. A workspace typically represents the entire production facility and consists of a background that is overlaid with radial menus and zoom regions. HMIWeb Display Builder is used to create a workspace and add the background, radial menus and zoom regions to the workspace. Collaboration Station is used to add contacts and other content to radial menus. For information, see “Configuring a workspace” in the *Collaboration Station User’s Guide*.

About dynamic shapes

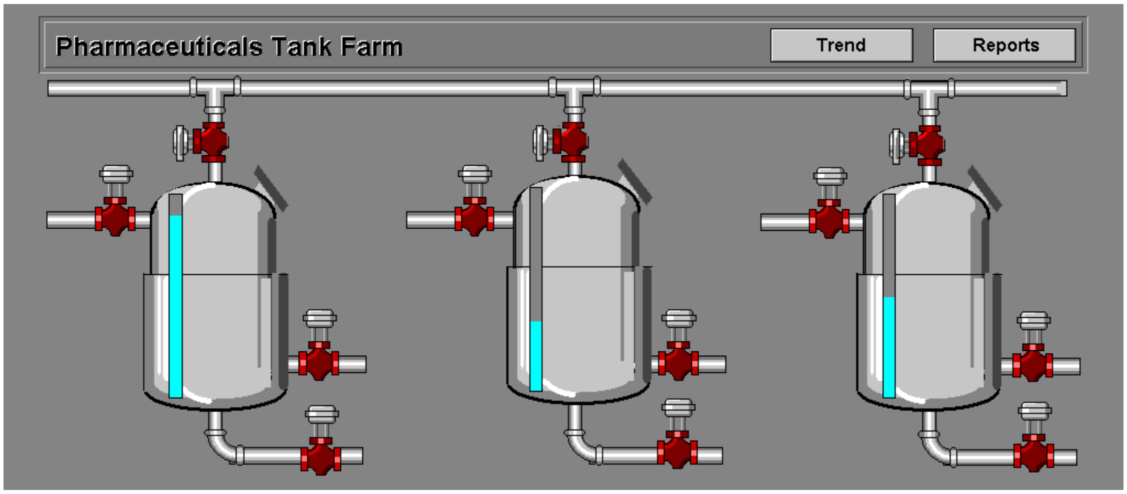
A custom object that can be used to present complex dynamic data.

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.



Related topics

"Creating a dynamic shape" on page 71

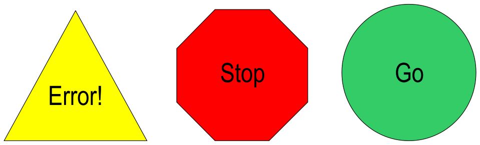
About shape sequences

A custom object that can be used as a status indicator or an animation.

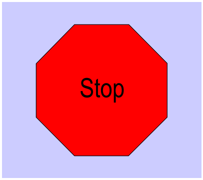
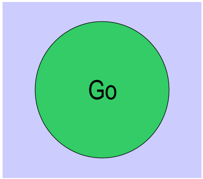
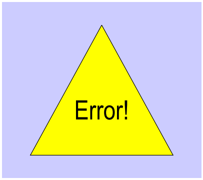
A *shape sequence* is, in effect, a 'custom object' that is used in displays as either a status indicator or an animation.

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	
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

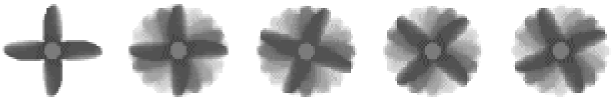
When a shape sequence is used with an analog point, the shapes are automatically allocated across the point's range (excluding the first shape if it represents a 'bad value'). For example, if the value ranges from 0 to 1000, and there are 11 shapes (the first representing a 'bad value'), then:

- 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.

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.)



Tip

Animations do not run in HMIWeb Display Builder. They can be viewed only in Station.

About custom properties

Describes how you can reuse dynamic shapes, shape sequences, and generic displays.

There are two types of custom properties, “*shape custom properties*” and “*system custom properties*.” The differences are in their scope and where they are stored.

The use of shape custom properties allows you to reuse dynamic shapes or shape sequences across multiple displays. You can map the custom property either when you insert the shape at design time or through scripting at runtime.

When using system custom properties, you can create a single generic display that can be reused across your plant. This is useful where the Asset model uses a consistent structure within the asset hierarchy.

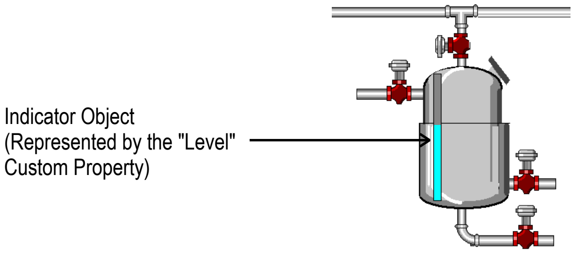
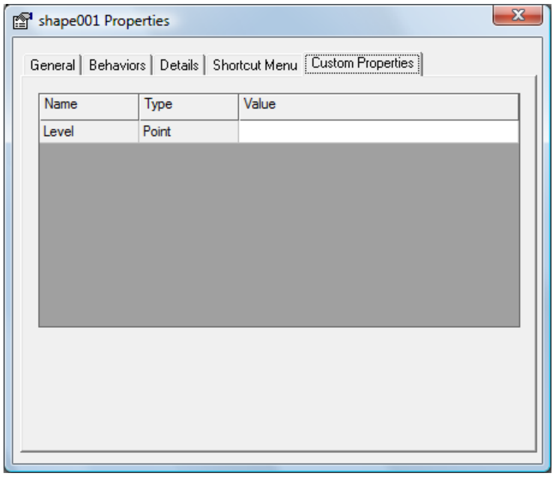
Using shape custom properties to map dynamic objects to points

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

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

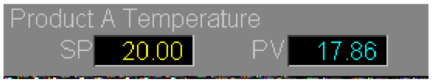
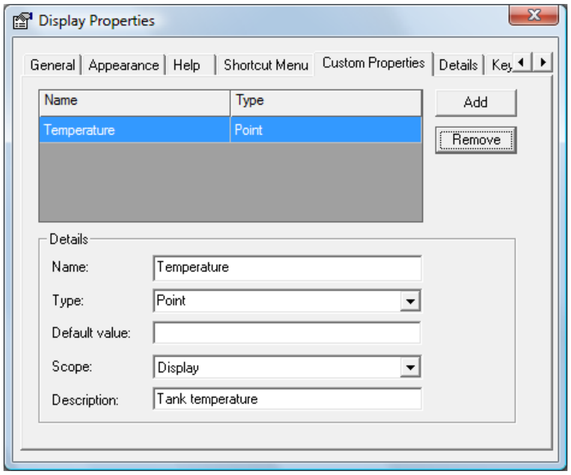
- The indicator is configured to show the PV of a point (which represents the level of liquid in the tank).
- The chart is configured to show the PV and SP of another point (which represent the tank's actual and desired temperatures)

The figure on the right shows how, when you insert this shape into a display at design time, you map the custom property to a point and parameter by typing its name and parameter in the **Value** column.

A typical dynamic shape	Points mapped to the custom properties when the shape is inserted into a display
 Indicator Object (Represented by the "Level" Custom Property)	shape001 Properties Name Type Value Level Point

Using system custom properties to map objects to points

When you insert an object into a display at design time, you use system custom properties to map the object to the appropriate points or 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)  	 Display Properties General | Appearance | Help | Shortcut Menu | Custom Properties | Details | Key Add Remove Name | Type Temperature | Point Details Name: Temperature Type: Point Default value: Scope: Display Description: Tank temperature

About popups and faceplates

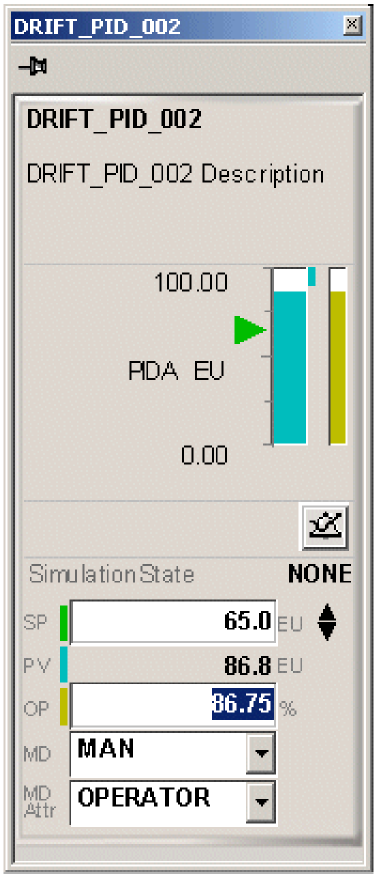
Describes popups and faceplates (a specialized type of popup) and their use in Station.

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.

Station can simultaneously display up to four faceplates. If a user clicks a fifth object, the oldest faceplate is replaced by the newest one.



DRIFT_PID_002

DRIFT_PID_002 Description

Simulation State NONE

SP 65.0 EU

PV 86.8 EU

OP 86.75 %

MD MAN

MD Attr OPERATOR

Figure 3: Typical faceplate

Migrating HMIWeb Display Builder faceplates

Describes how to convert pre-R310 faceplates to an improved version.

New functionality for faceplates was introduced with Experion R310. Faceplates that were created in versions earlier than R310 are standard HMIWeb displays based on the 'Normal Display' template. That is, they do not have auto-selection functionality enabled.

If your system has faceplates that were created prior to R310, you can convert them to faceplates in the current version so that auto-selection is enabled.

To migrate HMIWeb Display Builder faceplates

1. Open the pre-R310 display in HMIWeb Display Builder.
2. Choose **Tools** > **Convert to faceplate**.

   The display is converted to a faceplate display and a **Faceplate** tab is added to the **Display Properties** window.

   Note that this action does not resize existing displays to 150×325 pixels. It simply adds the components necessary to enable the auto-select functionality.
3. If the pre-R310 faceplate used scripts to implement auto-selection behavior:
   - a Remove the auto-selection scripts from the newly-migrated faceplate.
   - b Configure auto-selection properties on the **Faceplate** tab.
4. If the pre-R310 faceplate used scripts to handle certain commands, then that script may also not be required and you should remove it. Current versions of faceplates handle selection-independent commands without requiring script in the faceplate.

Related topics

"Creating a faceplate" on page 74

About generic displays

Explains how generic displays can be reused across your plant.

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.

Example: Creating a generic display for use across a plant

You want to create a generic display for three holding tanks. (Each time operators call up the display, they must specify the tank they want to view.)

You want the display to include an alphanumeric and an indicator that shows the level of the specified tank.

The tanks share a common location in the Asset Model, `/Assets/Train1/HoldingTanks`, and their *item names* are: `Tank1`, `Tank2`, and `Tank3`.

The level of each tank is represented by the PV of an item called: `Level`.

To achieve this you will need to:

- Create a new display, called `HoldingTanks.htm`. Then create a system custom property called `HoldingTank` with a scope of `Display`. (This custom property represents part of the *full item name* of the tank.)
- Add an alphanumeric object to the display and map it to the system custom property.
- Create a dynamic shape with a level indicator object, and assign a shape custom property called `TankLevel`. Then include the shape in display and map the shape custom property to the system custom property.

To create the display and 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`*.

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

To add the alphanumeric and map it to the custom property

1 Place an alphanumeric on the 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 your change to the display.

6 Test the display in Station by calling up the display, showing the PV for *Tank3*.

To call up the display in Station

- In the Command Zone in Station type:

  `HoldingTanks.htm?HoldingTank=Tank3`

  Because you specified *Tank3*, the alphanumeric is bound to PV of the point with the full item name of *\Assets\Train1\HoldingTanks\Tank3\Level*.

To create the dynamic shape

1 Open *`3dtank04.sha`* from the shape library.

Shape libraries can be found in *`<data folder>\Honeywell\Experion PKS\Client\HMIWeb Display Builder\ShapeLib`*.

Where *`<data folder>`* is the location where Experion data is stored. For default installations, *`<data folder>`* is *`C:\ProgramData`*. The *`C:\ProgramData`* folder is a system folder, which means that it is only visible if you select the **Show hidden files, folders, and drives** option button in the **Folder Options** dialog box. To change this setting in Windows Explorer, click **Organize** > **Folder and search options**, and then click the **View** tab.

2 Save it as *`TankLevel.sha`*.

Honeywell strongly recommends that you do not change a supplied shape, so that your changes are not overwritten during an upgrade. Instead, save the shape with a new name, and then make the required changes.

3 Without anything selected, double-click the shape.

The shape **Properties Window** opens.

4 Click the **Custom Properties** tab.

5 Click **Add**.

6 Type `TankLevel` in the **Name** box.

7 Double-click the **Type** box and select *`Point`*.

8 Close the Shape Properties.

9 Add an indicator object to the tank.

10 Double-click the indicator object.

The **Properties Window** for the indicator object opens.

11 Click the **Data** tab.

12 From the Point list, select **<TankLevel>**.

13 Group the objects.

14 Save and close the shape file.

To insert the shape and map the shape custom properties

1. Open *`HoldingTanks.htm`*, which 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 upper-left of this rectangle marks the upper-left corner 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 \Train1\HoldingTanks\<HoldingTank>\Level*.
7. Close the Properties Window.
8. Save the display.

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

To call up the display in Station

- In the Command Zone in Station type:

  ```
  HoldingTanks.htm?HoldingTank=Tank3
  ```

  Because you specified *Tank3*, both the alphanumeric and indicator are bound to the point with the full item name of *\Assets\Train1\HoldingTanks\Tank3\Level*.

About Pan and Zoom displays

A *Pan and Zoom* display is a custom display which can be the size of several normal displays, allowing you to author a single large display covering all, or a large portion, of your process in a single display. An example of a display that would benefit from this functionality is a map of a long pipeline with equipment at various locations.

When an Operator views a Pan and Zoom display in Station, a *thumbnail* provides a representation of the entire display as well as an indication of the current “viewport”. The *viewport* indicates which portion of the overall display you are currently viewing.

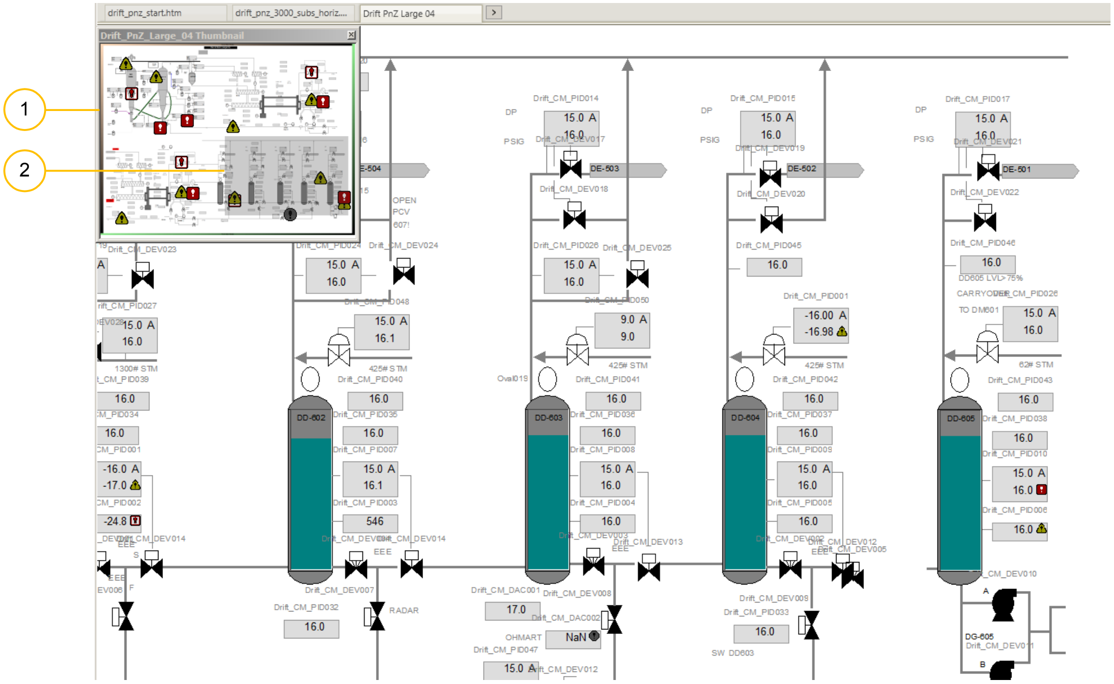


Figure 4: Pan and Zoom display, showing thumbnail, viewport indicator, and active alarms

Table 1: Pan and Zoom display elements

Item	Description
1	*Thumbnail* Provides an overview of the entire display as well as an indication of the current viewport.
2	*Viewport* Indicates which portion of the overall display you are currently viewing.

The thumbnail works as a navigational aid within the larger display, and also contains icons to indicate active alarms from the entire display, both inside and outside the current viewport.

When viewing a Pan and Zoom display you can *pan* to different parts of the overview display, and *zoom* in and out of the display. As well as being able to zoom within a display, it is also possible to create targets in a display that, when clicked, invoke another display at a pre-defined zoom level. This function is referred to as *hyperzoom*.

About Collaboration Station Workspace displays

Collaboration Station is a licensable option of Experion. A Collaboration Station is a type of Experion Station that presents your operation on a large interactive touchscreen in a view designed to facilitate communication and collaboration.

Collaboration Station is highly flexible and can be customized to meet the collaboration requirements at your facility. It is intended for use whenever two or more people need to discuss operational matters and share relevant material. This might include operational meetings such as shift handovers, troubleshooting meetings to diagnose and solve problems, and facility showcases.

Collaboration Station integrates with the Microsoft Lync Unified Communications platform to enable remote collaboration.

Collaboration Station can be used to gather a range of key content so that it is easily accessible during a collaboration. This can include Experion displays, documents, websites, and other common file types.

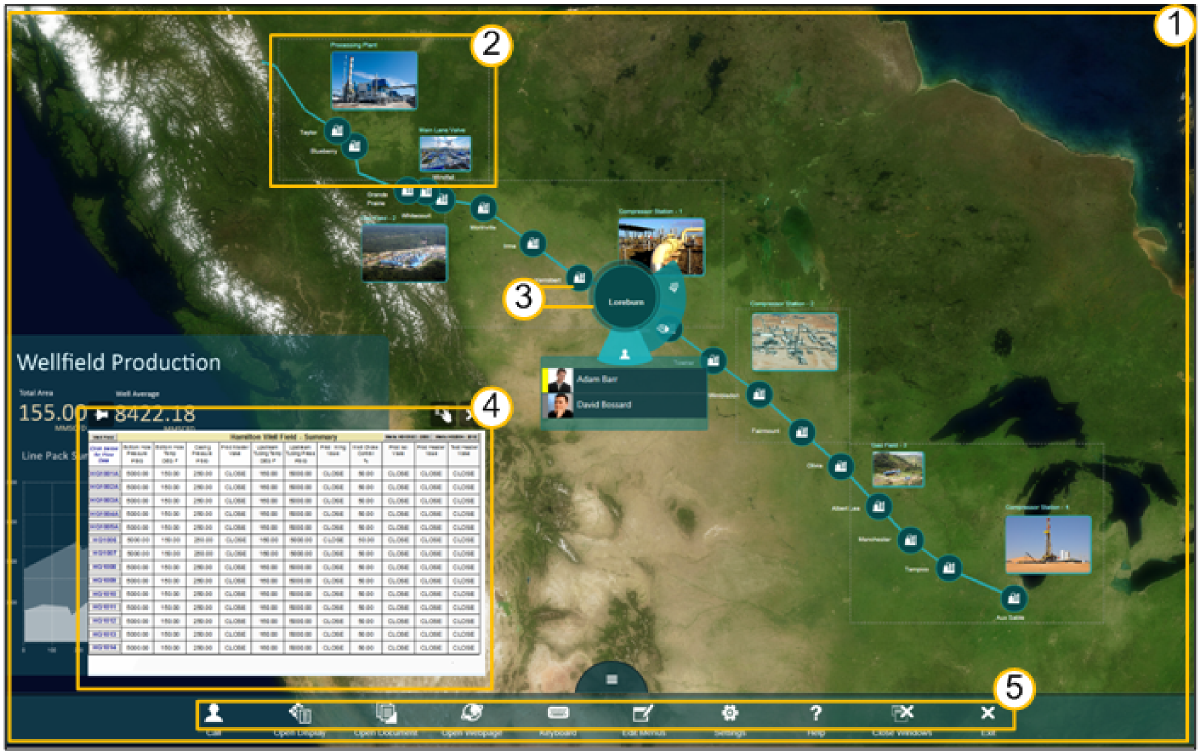


Figure 5: Layout of a Collaboration Station workspace

Item	Description
1	*Workspace* A workspace is a display created in HMIWeb Display Builder that represents the facility. It consists of static images, zoom regions, radial menus, dynamic content such as Experion trends, alphanumerics, alarm icons, and solution pack shapes. Workspaces can be created only on Collaboration Station nodes.
2	*Zoom region* Zoom regions are rectangular and are used to define the areas on the workspace that can be enlarged with a single tap. They can be used as a placeholder for open content windows. Content windows opened after zooming into a region are made small after zooming out and enlarged to their original size when returning to the zoom region.

Item	Description
3	*Radial menu* Radial menus provide access to content and Microsoft Lync contacts. For more information about what content can be accessed from Collaboration Station, see “Software compatibility” in the *Collaboration Station User’s Guide*. If a radial menu is associated with an asset or location, it will display alarms for the asset or location.
4	*Content window* A new window is opened when content is accessed via a radial menu or the workspace toolbar.
5	*Workspace toolbar* When opened, the workspace toolbar provides access to help, settings, content that is not available via a radial menu, and radial menu edit mode.

About Equipment displays and templates

Quick Builder includes functionality that enables you to create equipment items based on templates. As part of that process, other items such as points and controllers that would typically be created individually in Quick Builder are generated automatically saving considerable engineering effort.

The Equipment Template Builder is a tool in which you create and modify equipment templates. When using templates to facilitate creating equipment items, part of the process is to create HMIWeb schematics to be included in the detail displays for that equipment. These schematics (including trends) then appear embedded within the Equipment Detail display in Station.

Creating schematics for use in equipment detail displays

1. Create a standard display.

   Tip

   Equipment displays are different to typical Station displays in that they have extra display artifacts such as navigation bars and banners. These items will therefore need to be considered as they will impact available real estate and the overall aspect ratio of the finished display. See the *Quick Builder User's Guide* for more information.

2. Add elements to the display as required.
3. Open the **Properties** window.
4. Click the **Data** tab and for each item you want to collect data for:
   - Set the Point value to *`<CurrentPoint>`*
   - Set the Parameter value to *`<Equipment_parameter_name>.Parameter`*, where *`<Equipment_parameter_name>`* is the equipment name defined in the Equipment Template Builder, and *`Parameter`* is the parameter you want to capture for that equipment item, such as `PV` or `SP`.

     For example, *`meter_volume_flow.SP`* collects SP data for the equipment point *`meter_volume_flow`*.

5. In the Equipment Template Builder on the **Views** page for the relevant template, create a view for this new schematic placing its address in the **URL** field.
6. In the **Value** field, define the value for **CurrentPoint** as `CurrentPoint=[%TagName%]`. For example, *`sysdtlwell.htm?CurrentPoint=[%TagName%]`*

   When the template is imported into Quick Builder, the Equipment Detail display is updated to include the new schematic.

For more information about creating and configuring Equipment displays, see the *Quick Builder User's Guide*.

About archive format

A way to save a display or shape in a single file.

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.

Archive format is useful if you want to E-mail a display to someone.

Saving a display/shape in 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.

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**.

Saving a display/shape in both archive and standard format

To save a display/shape in both archive and standard format

- Select **Tools** > **Options** and select **Save unarchived copy of file with archive files**.

About display definition files

Details of the files generated automatically for each display.

When you save a display, HMIWeb Display Builder automatically generates other files to support the *.htm* display file. These files are stored in a folder that is created and maintained by HMIWeb Display Builder in the same location as the *.htm* display file. The folder name is the display file name, minus the .htm extension, and appended with *_files*. This folder is often referred to as the display's "underscore" folder.

For example, if the display name is *Plant West.htm*, the "underscore" folder name is *Plant West_files*.

File	Description
bindings.xml	An XML file that contains the data binding definitions for the objects on the display.
*_datasource1.dsd	An XML file that contains the property definitions for the objects on the display.
*.cnf.xml	An XML file that contains the alarm group definition for the display. This file is only generated when the **Generate an alarm group file when saving a display** check box is selected in the HMIWeb Display Builder options.

Attention

These files are automatically generated by HMIWeb Display Builder based on the display properties and object definitions that you create within the display. You should not edit any of these files.

In addition to these files, other files may also be stored within this "underscore" folder, such as shapes and images.

Associating a custom display with an alarm group

If you associate a custom display with an alarm group, an icon will appear on the tab for that display when its group contains an active alarm.

If you have enabled tabbed displays, you need to associate the custom display with an alarm group if you want an alarm icon to appear on the tab of that display to indicate an active alarm in that alarm group.

You associate a custom display with an alarm group using the display object's **Associate with an alarm group** property.

About alarm group names

When you associate an alarm group with a custom display, you can specify the name of a custom alarm group, or you can use the default alarm group name generated by HMIWeb Display Builder. The default alarm group name is customizable.

Creating alarm groups

Associating a custom display with an alarm group does not generate that alarm group within the system. Either the alarm group already exists within the system, or you enable HMIWeb Display Builder to generate the alarm group file when the display is saved, which you then need to import into the system using Enterprise Model Builder.

Changing the default alarm group name

The default alarm group name associated with a display is based on the alarm group naming convention.

This default alarm group name is *<DisplayName>_almgrp*, where *<DisplayName>*is replaced with the actual name of the display. You can change the default alarm group naming conventions in HMIWeb Display Builder.

Tip

Within the display properties, you can specify whether to associate the display with the default alarm group, which is based on the alarm group naming conventions described here, or to associate the display with a custom alarm group name. This procedure does not change the custom alarm group name.

To change the default alarm group name

1 Choose **Tools** > **Options**.

2 Click the **Alarm groups** tab.

3 In the **Default alarm group naming convention** box, type the default alarm group name to use.

Attention

You can use the *<DisplayName>* variable within the default alarm group name, which is replaced with the actual display name when you save the display.

4 Click **OK**.

Enabling HMIWeb Display Builder to generate alarm groups

To generate an alarm group file when a display is saved, you need to enable this in HMIWeb Display Builder. After the alarm group file is generated, you can then import this into the system using Enterprise Model Builder.

To enable HMIWeb Display Builder to generate alarm groups

1 Choose **Tools** > **Options**.

2 Click the **Alarm groups** tab.

3 Select the **Generate an alarm group file when saving a display** check box.

4 Click **OK**.

Generating alarm group files for multiple displays

You can generate alarm group files for display files contained within a folder, or you can generate a single alarm group file that contains the points for all of the display files within a folder.

Instead of opening and then saving existing display files to generate an alarm group file, you can use the Display Builder Assistant to generate the alarm group files for multiple display files.

To generate alarm group files for existing display files

1 Choose **Tools** > **Display Builder Assistant**.
The **HMIWeb Display Builder Assistant** window appears.

2 In the **Task** list, select **Generate Alarm Groups**.

3 Next to the **Display folder** box, click **Browse** and locate the folder that contains the display files for which you want to generate alarm group files. If you want to generate alarm groups files for display files contained in sub folders of the selected folder, select the **Include subfolders** check box.

4 Do one of the following:

Option	Description
To generate an alarm group file for each display contained in the selected folder.	1. Clear the **Generate single Alarm Group file** check box. 2. Next to the **Save Alarm Group file to** box, click **Browse** and locate the folder where the alarm group definition file will be saved, and then click **OK**. **Attention** By default **Alarm Group** folder will be selected to save the alarm group definition files Select the **Save alarm group files to** option to save the alarm group definition files to somewhere other than the `_files` folder for each display.
To generate a single alarm group file that contains the points for all of the display files within a folder.	1. Select the **Generate single Alarm Group file** check box. 2. Next to the **Save Alarm Group file to** box, click **Browse** and locate the folder where the alarm group definition file will be saved, and then click **OK**. If you select only the folder, the alarm group definition file will be saved as `BulkAlarmgroupfile.cnf.xml`. If you want to specify another file name, after the path displayed in the **Save Alarm group file to** box, you can type a backslash (\) followed by the file name and file extension, for example `filename.cnf.xml`. If the file extension is not given, then it will be used as a folder name.

5 Click **Run**.

Creating a new display

When creating a new display, note that:

- 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.
- The default folder for custom displays is `<data folder>\Honeywell\Experion PKS\Client\Abstract`.

  Where `<data folder>` is the location where Experion data is stored. For default installations, `<data folder>` is `C:\ProgramData`. The `C:\ProgramData` folder is a system folder, which means that it is only visible if you select the **Show hidden files, folders, and drives** option button in the **Folder Options** dialog box. To change this setting in Windows Explorer, click **Organize** > **Folder and search options**, and then click the **View** tab.

  Tip

  Honeywell recommends copying all custom displays to the *abstract folder* as a general best practice, as system displays will not load correctly if they are on a network share.

- When you save a display in standard format, an associated folder is created that contains various support files and graphics. The folder has the same name as the display, but with a `_files` extension. If you remove a shape or a picture from an HMIWeb display, the shape or picture file is removed from the `_files` folder when the display is closed in HMIWeb Display Builder. This ensures that the `_files` folder does not get filled with content that is no longer referenced.

Related topics

"Object count limits" on page 64

"Creating a standard display" on page 65

"Creating a Pan and Zoom display" on page 66

"Creating a Collaboration Station Workspace display" on page 68

"Creating a display from a Web page" on page 70

"Creating a dynamic shape" on page 71

"Creating a shape sequence" on page 72

"Creating a popup" on page 73

"Creating a faceplate" on page 74

"Creating a point detail display" on page 75

"Process Point Parameter" on page 76

"Container Point Parameter" on page 77

Object count limits

When creating displays you need to be aware of the count limits for objects on a display. Exceeding these limits can cause a display to fail. Examples of a display failing are:

- Script errors when the display is loaded.
- Objects not appearing when the display is loaded.
- Unpredictable behavior when the display is loaded and subsequently reloaded.

As you are creating a display it is a good idea to use the Display Performance Analysis tool to check that the object count limits are not exceeded. The Display Performance Analysis tool lists the limits for each type of object as well as the limit for the total number of objects on a display. These limits are recommended for performance; however, there are scenarios where exceeding these limits can cause errors on a display or cause the display to malfunction.

Related topics

“Using the Display Performance Analysis tool” on page 140

Creating a standard display

Tip

To speed up your work and to ensure visual consistency, you should create an appropriate set of templates and style sheets.

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.
4. Add objects to the display as required.

Related topics

"GetValue method" on page 447
"PutValue method" on page 460

Creating a Pan and Zoom display

To create a Pan and Zoom display

1. Either:
   - a Click the arrow to the right of and choose **Display from Template**, or
   - b From the menu bar, select **File > New > Display from Template**
2. Select the **Pan and Zoom Display** template and click **OK**.
   A new display appears.
3. Configure the display's properties as appropriate.
4. Add objects to the display as required by adding elements from the tool bars or by copy and pasting existing content from other displays. Note that when copying content from other displays only the element level scripts are copied. No page level scripts nor anything in the general section is copied.

   Tip

   The default display size, as shown on the **Appearance tab**, is 3200 x 2400. To identify the appropriate size for your Pan and Zoom display, first add objects to the display and when you have finished turn on the ruler to determine the size of the completed display. Enter these size details on the Appearance tab in step 7.
5. If required, the default Pan and Zoom SafeView category can be changed for this display. It may be that you want all your Pan and Zoom displays to have the same category, making them easier to manage within SafeView.
6. Re-size the display by selecting **Display properties > Appearance > Size** to reflect the content that has been added to the display. The overall size should be larger than the resolution of the target monitor to facilitate panning.
7. Save the new display.
   The Pan and Zoom display is saved and a thumbnail is created.

   Attention

   1. Not all content from the main display is also displayed in the thumbnail. The thumbnail will not contain any scripts or dynamic objects other than alarm icons.
   2. Honeywell recommends that you copy all Pan and Zoom displays to the `<install folder>\ProgramData\Honeywell\Experion PKS\Client\Abstract` directory,, where `<install folder>` is the location where Experion is installed. If the displays are stored anywhere other than the local file syste, (such as on a mapped drive), they will not load correctly.

Converting existing displays to a Pan and Zoom display

1. Identify the displays you want to combine into one Pan and Zoom display.
2. Create a new pan and zoom display, as described in the previous procedure .
3. Copy the content from each display and paste onto the new Pan and Zoom display.
4. Ensure that you remove any duplicates of script holder shapes on the final Pan and Zoom display.
5. Save the display.

   Tip

   Only scripts for display elements are copied across to the new display. If any page-level scripts exist on the 'old' display, they will not be copied across so you will need to do this manually.

Adding hyperzoom to a Pan and Zoom display

Hyperzoom is a way to jump from a target in one Pan and Zoom display to a pre-defined destination in another Pan and Zoom display. The destination Pan and Zoom display can also be invoked to a pre-defined zoom level.

Hyperzoom is configured through a URL parameter, and can be configured in a pushbutton or hyperlink, or entirely from script, for example *window.external.currentPage = "Unit3Overview.htm? pnzanchor=tank1&pnzzoom=1.5"*.

To configure hyperzoom

1 Select the item, either a push button or a hyperlink, to be configured with hyperzoom.

2 In the **Button Details** area of the **Property Grid**, select `Callup Page` from the **Action** list.

3 Type the following in the **Callup Page** field:

```
view htm?pnzanchor=<name>&pnzzoom=<zoom level>
```

where, <*name*> specifies the element ID on the destination display that the viewport will centre around, and <*pnzzoom*> specifies the zoom level. Valid values are between `0.8` and `2.0` where `2.0` is equivalent to 200% zoom.

Tip

- The *pnzzoom* parameter is optional. If not specified, the zoom level defaults to 100%.
- If the specified *pnzanchor* cannot be found on the destination, page the hyperzoom is ignored and the display is invoked as normal (meaning the *pnzzoom* is also ignored)
- If the specified *pnzanchor* is valid but the *pnzzoom* is outside the valid range (for example, `pnzzoom=5`) it is converted to a valid value. In this example, the value would be changed to 2.0 (200% zoom)
- If the specified *pnzanchor* is valid but the *pnzzoom* is not, for example `pnzzoom=abcd` (it cannot be converted to a number between 2.0 and 0.8), the display is invoked centered around the pnzanchor but at 100% zoom

Creating a Collaboration Station Workspace display

Before creating a Collaboration Station Workspace display, you should consider the design of the Workspace. For more information, see “Collaboration Station design” in the *Collaboration Station User’s Guide*.

After creating a Collaboration Station Workspace display, you must configure the radial menus to include the required content. The content can include Experion displays, contact lists for remote collaboration, webpages, remote applications, and documents. For more information, see “Configuring a radial menu” in the *Collaboration Station User’s Guide*.

Prerequisites

- You have identified the images to be used in the background and within radial menus.
- You have identified any dynamic content to be embedded in the background, such as trends, alarm icons, and solution pack shapes.

To create a new workspace display

1 On the Collaboration Station node, choose **Start > All Programs > Honeywell Experion PKS > Client Software > HMIWeb Display Builder**.

2 On the **File** menu, click **New** and choose **Display from Template**.

3 Select **Collaboration Station Workspace**.

4 Click **OK**.
A new display, based on the Collaboration Station Workspace template, appears.

5 In the HMIWeb Display Builder properties grid, set the workspace properties:

> **Tip**
> For the correct operation of Collaboration Station, it is recommended that the workspace background image has an aspect ratio of 16:9 and that the pixel resolution is the same as the resolution of the Collaboration Station computer screen. For example, 1920 x 1080.

a Use the **Image** property to assign a an image file to be shown as the background.

b Use the **Size** properties to assign a pixel height and width to the workspace.
It is recommended that the workspace is the same size as the background image.

6 Add objects to the workspace display as required.

7 Save the workspace display.
The default folder for Collaboration Station workspaces is `<data folder>\Honeywell\Experion PKS\Client\Abstract`. Where `<data folder>` is the location where Experion data is stored. For default installations, `<data folder>` is C:\ProgramData. The C:\ProgramData folder is a system folder, which means that it is only visible if you select the **Show hidden files, folders, and drives** option button in the **Folder Options** dialog box. To change this setting in Windows Explorer, click **Organize > Folder and search** options, and then select the **View** tab.

To add a zoom region

1 In HMIWeb Display Builder, create or open a workspace.

2 On the **View** menu, click **Shape Gallery**.
The Shape Gallery appears.

3 In the Shape Gallery Navigation Pane, browse to and click **Collaboration Station**.
The objects available to add to a workspace are displayed.

4 Drag `zoom_region` to the required location on the workspace.

5 In HMIWeb Display Builder, resize the rectangle as required.

Tip

It is recommended that you make the zoom region no larger than 200 x 110 pixels so that Experion displays opened after zooming in to the region will be automatically unsubscribed from Experion process data after zooming out.

To add a radial menu

1. In HMIWeb Display Builder, create or open a workspace.
2. On the **View** menu, click **Shape Gallery**.

   The Shape Gallery appears.
3. In the Shape Gallery Navigation Pane, browse to and click **Collaboration Station**.

   The objects available to add to a workspace are displayed.
4. Drag `radial_menu` to the required location on the workspace.
5. In the HMIWeb Display Builder properties grid, set the radial menu properties:

   a. Use the **AssetName** custom property to assign a name to be displayed on the radial menu. If the name is an Experion asset or point, the radial menu will display alarms associated with the asset or point. The same asset or point can be associated with more than one radial menu. All radial menus on workspaces on the same Collaboration Station node that are associated with the same asset or point, share the same content. For example, they all contain the same contacts, Experion displays, and documents. Radial menu content is not shared by different Collaboration Station nodes.

   b. Use the **AssetIcon** custom property to associate the radial menu with an image file to be shown on the radial menu. The standard radial menu images can be found on the Collaboration Station computer in `c:\Program Files (x86)\Honeywell\Experion PKS\client\Station\HMIWebJS\CollaborationStation\images\RadialMenu\StandardAssetIcons`.

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.

Attention

If you use the zoom function in HMIWeb Display Builder or Station, scroll bars 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

Before creating a dynamic shape, it is important to consider the following:

- 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'.
- All objects within a dynamic shape must be grouped to form one object.
- You should store the shape in a registered folder.

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.

3 Add objects to the shape as required.

4 Assign custom properties to the dynamic objects, as appropriate.

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.

To assign the custom properties

1 Select a dynamic object and click the **Properties** 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 the above steps for the remaining dynamic objects that you want to map to shape custom properties.

Related topics

"About dynamic shapes" on page 44

A custom object that can be used to present complex dynamic data.

"GetCustomProperty method" on page 430

"SetCustomProperty method" on page 472

Creating a shape sequence

When creating a shape sequence you must observe the following guidelines:

- 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'.
- You should store the shape sequence in a registered folder.

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.
3. Add shapes (objects) as required.

Related topics

“General (Shapes) properties” on page 251

Creating a popup

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 size of the popup, which is 300×300 by default.)

3 Add objects to the popup as required.

Creating a faceplate

Because Experion includes faceplates for all point types that are typically used in an Experion system, you only need to create a faceplate if you have custom points.

The file name 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 file name of the point detail display for a particular type of container point is *`BoilerType1.htm`*, the file name of the faceplate must be *`BoilerType1_fp.htm`*.

To create a faceplate

1 Choose **File** > **New** > **Display from Template**, click **Faceplate Display** and then click **OK**.

Alternatively, click the arrow to the right of 🗔 and choose **Popup**.

2 If necessary, you can change the faceplate's size from the **Appearance** tab on the Properties window. (The dotted lines indicate the default size, which is 300×300 pixels.)

3 To configure the page-level display properties, go to the Properties window, click the **Faceplate** tab and then specify the elements any auto-selection behavior.

4 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`*.

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

6 Save the faceplate with the appropriate filename (including the '*`_fp`*' suffix).

Related topics

"Migrating HMIWeb Display Builder faceplates" on page 49

Describes how to convert pre-R310 faceplates to an improved version.

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[1]	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
Data entry allowed[1]	Allows users to change the parameter's value. If you allow data entry, select the minimum **Security Level** required to change the value.

[1] Not applicable to a chart.

Container Point Parameter

Property	Description
Type of database link[2]	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[2]	Allows users to change the parameter's value. If you allow data entry, select the minimum **Security Level** required to change the value.

[2] Not applicable to a chart.

Drawing techniques

This section 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 buttons on the **Toolbox** toolbar or on the **ToolBox** tab to create objects.

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

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

1 Click the appropriate object button.

Clicking this…	**…creates**
■	Rectangle
▢	Rounded rectangle
⬬	Oval
◜	Arc
◢	Wedge
☑	Check box
▭	Pushbutton
▮	Indicator
▤	Combobox
📈	Trend
📉	Basic trend

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 at a 45–degree angle, 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 SHIFT + ENTER.

> **Attention**
>
> Perform a test printout to check that the textbox width is adequate for the expected content length. In some cases, the printout of the display may truncate the text content if the textbox is too narrow. If necessary, increase the width of the textbox slightly (usually only a few pixels).

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.

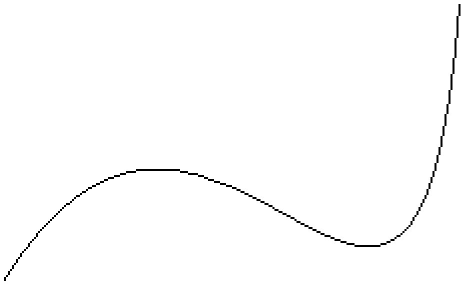
By default, hyperlinks are blue and underlined, but you can change this if appropriate.

4 While the object is still selected, specify the link’s target:

Method	Description
Properties tab	1. Click the **Properties** tab. 2. Expand the **Details** section. 3. Specify the link's target, either a Web page or another display.
Properties window	1. Click 📝 on the **View** toolbar. 2. Click the **Details** tab. 3. Specify the link's target, either a Web page or another display.

Creating a Bezier curve

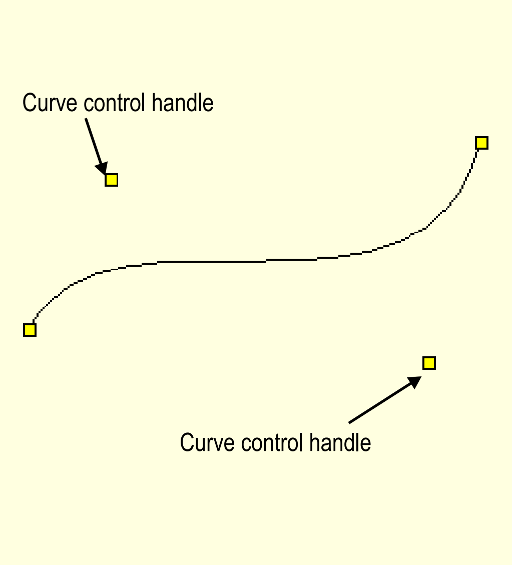
You use the **Bezier** tool—∿—to create a *Bezier curve* (a type of line), such as the one shown in the following figure.

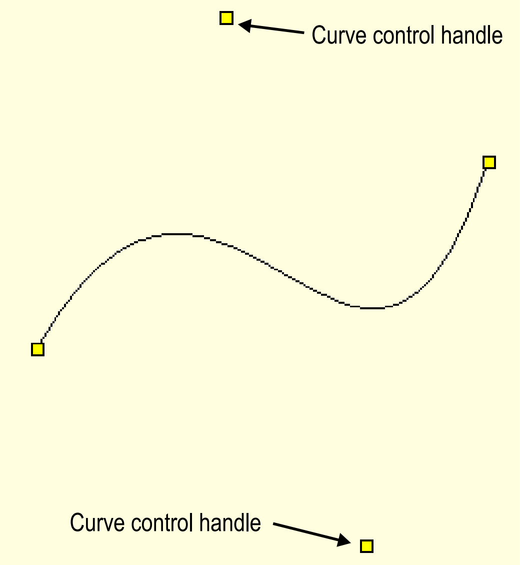


A Bezier curve has *curve control handles* that control the path the curve takes between nodes. The following curves, which have the same start and end points, show how a curve control handle determines:

- 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





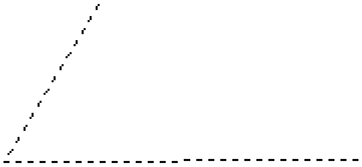
To create a Bezier curve

1 Click ∿ on the Toolbox toolbar.

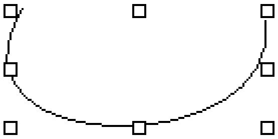
2 Click in the display to mark the first node (start of the curve).

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).



Creating a new object from overlapping objects

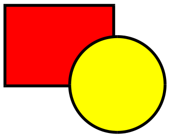
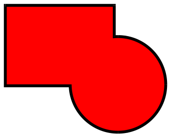
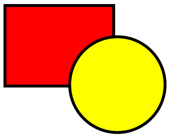
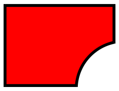
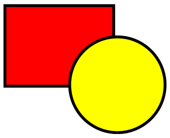
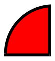
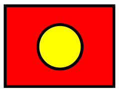
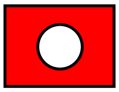
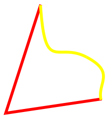
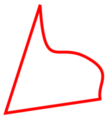
You can create a new vector graphic object from two or more overlapping vector graphic objects. The following table shows typical results—note, however, that the results depend on the object types and the way they overlap. When creating a new object from overlapping objects, please consider the following:

- 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.

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 red object is selected first):	And click this:	You get this:
		
		
		
		
		

You can only use 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

- 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 the SHIFT key 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.

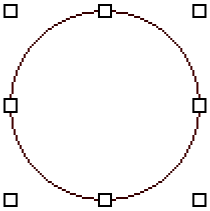


Figure 6: A selected object

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

- Use the following table

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

When duplicating an object(s) consider the following:

- 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 and unlocking 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 gray to indicate that the object is locked.

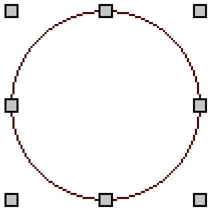


Figure 7: The selection handles of a locked object

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

To unlock an object

1. Select the object.
2. Click 🔒 on the Arrange toolbar. Note that the selection handles are no longer gray, and the object can be modified with the mouse.

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.

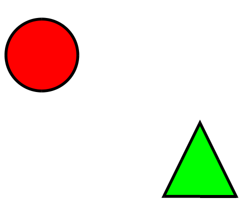
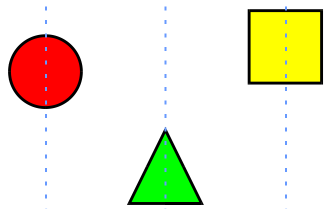
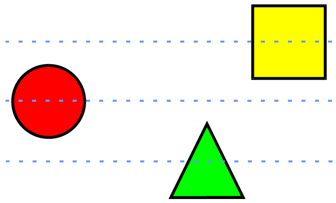
To move the objects:		Click this:
Horizontally so that:	Their left edges are aligned to the left edge of the reference object	
	Their centers are aligned to the center of the reference object	
	Their right edges are aligned to the right edge of the reference object	
Vertically so that:	Their tops are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

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.

If you find that a group does not resize correctly when zooming the display (this is more likely to affect small objects in a group), you can combine the objects.

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 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

- Select the group object and then click ⧉.

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.

> **Attention**
>
> 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 ⊞ group002 to this group002.

To uncombine a previously combined object

- 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 text box, graphic, and vector graphic objects.

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

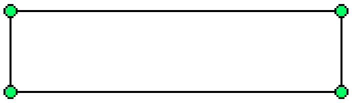
To rotate one or more objects 90 degrees

1. Select the object(s).
2. Click the appropriate button on the Arrange toolbar.

To rotate the object(s):	Click this:
90 degrees counterclockwise	↺
90 degrees 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 counterclockwise. 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.

Tip

Extra tabs may appear when you select a behavior— use them to define the behavior's properties.

To define a behavior

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

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).

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

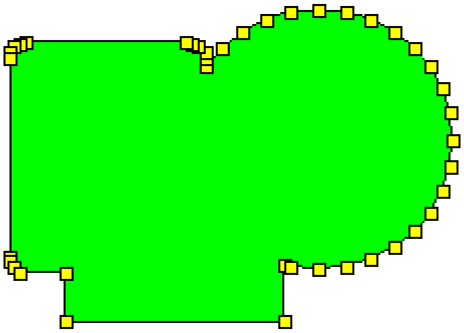
- Select the object you want to convert and click on the Transform toolbar.

Editing an object's nodes

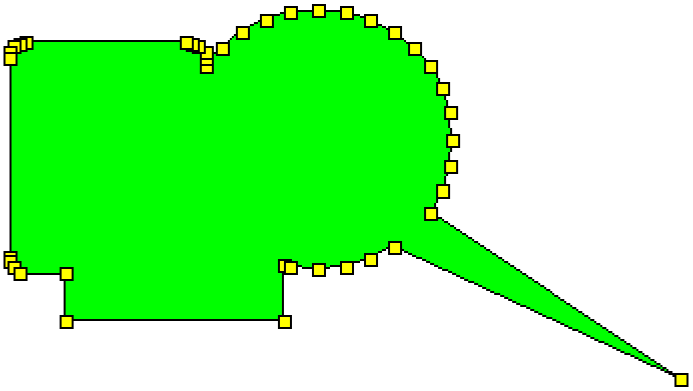
To edit an object's nodes

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.



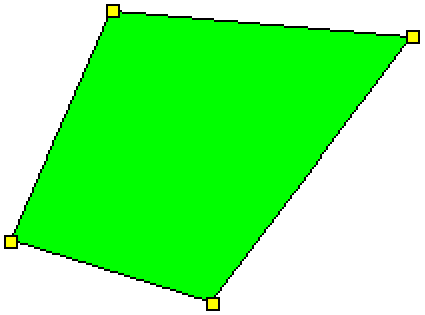
Adding a Node

You can also change an object's shape by adding a node between two existing nodes. You can only add nodes to polygons or polylines. You cannot add nodes to objects that have been converted to a path.

To add a node

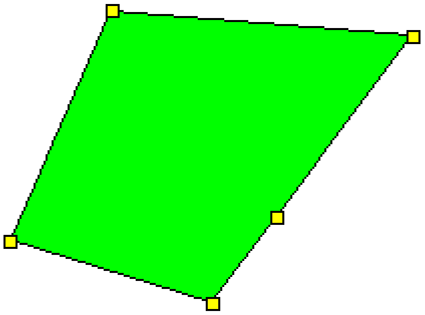
1 Select the object.

2 Click on the Toolbox toolbar.

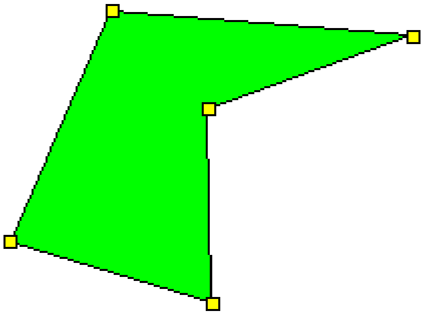


3 Move the mouse pointer over the line between two existing nodes.

4 Click when the pointer turns into a 'crosshair'—this adds the new node.



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



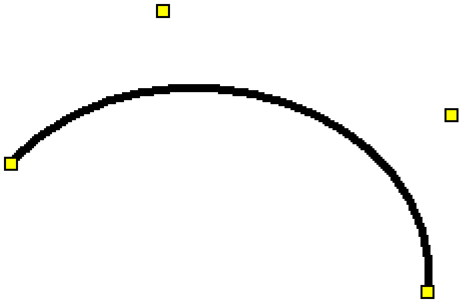
Editing an object's curve control handles

Some types of object, such Bezier curves, have curve control handles that control the path the line takes between nodes.

To edit an object's curve control handles

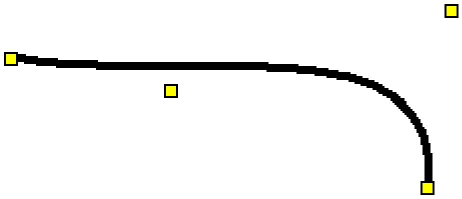
1 Select the object.

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.

To move an object a small distance

- Select the object you require and press the arrow key relating to the direction you want the object to move in.

  The object will move one pixel in that direction if the snap-to-grid function is off, or one grid unit if it is on.

Tip

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.

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.

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

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.

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. 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*)	
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.

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

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.) If you are linking to a shape file, save the display before you insert the shape so that the shape is linked with a relative path. (Inserting a linked shape before the display has been saved inserts the shape with an absolute path, which causes problems if the shape files are not distributed to remote Stations.)

You cannot insert a linked shape sequence unless you have registered the folder that contains it.

You can also insert a shape sequence by dragging the shape sequence from the Shape Gallery.

To insert a shape sequence

1. Open the display.
2. If the display has not already been saved, save the display to ensure that linked shape files are inserted with a relative path.
3. Click on the Toolbox toolbar.
4. 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.
5. 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.
6. If necessary, move and resize the shape.
7. Open the Properties Window, click the **Behaviors** tab and select **Shape Sequence Animation**.

   The Data and Animation tabs are added to the Properties Window.
8. 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.
9. 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 shapes 2 to 5.)

Related topics

"Registering a shape folder" on page 273

Inserting a dynamic shape

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.)

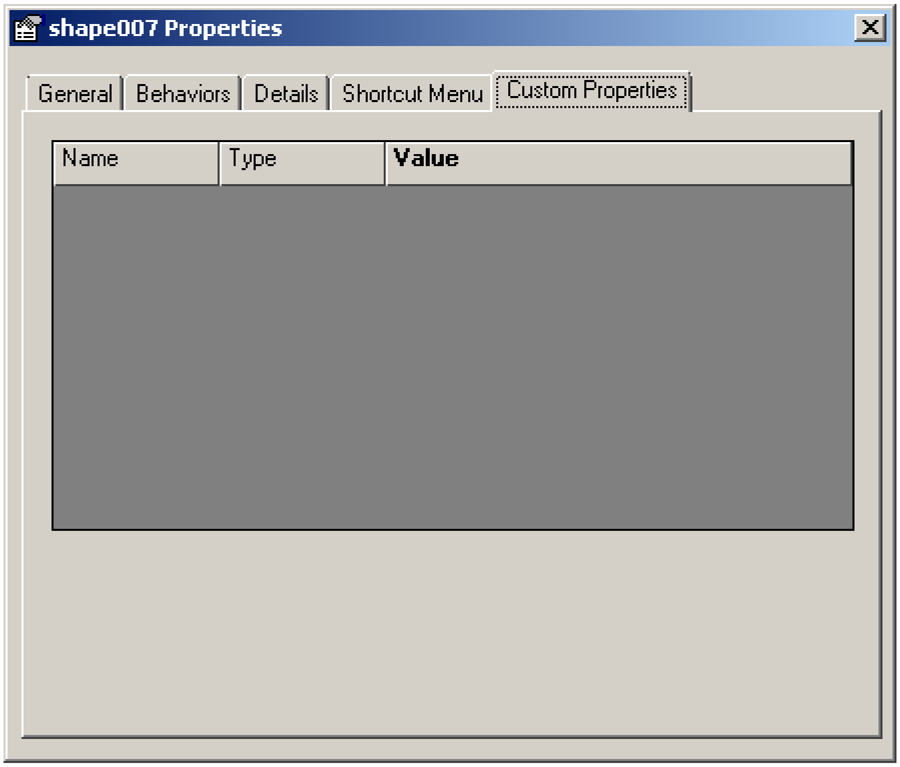
You cannot insert a linked dynamic shape unless you have registered the folder that contains it.

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**.



Name	Type	Value

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.)

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.

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.

The following procedure only works if the shape file is in a registered folder.

You can use the HMIWeb Display Builder Assistant to upgrade embedded shapes in every display within a specified folder.

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

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.

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 operators.

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.

Related topics

"Using ActiveX controls in displays" on page 117

Inserting an ActiveX document

You can insert an ActiveX document into a 'window' that is embedded into a display. (Scroll bars 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).

Inserting an alarm state icon

You can use an alarm state icon in two ways:

- To show the alarm state of the associated point (this is the way in which the icon is normally used)
- To show an 'alarm state' that is controlled by a script

Note that, at runtime, an alarm state icon is invisible if there is no alarm and the point is not disabled.

To insert an alarm state icon that shows a point's alarm state

1. Click ⚠ on the Toolbox toolbar.
2. Move the pointer to the display and drag it diagonally to mark the position of the alarm state icon. (The top-left of the rectangle marks the top-left of the alarm state icon.)
3. Double-click the alarm state icon to open the **Properties Window**.
4. Click the **Data** tab.
5. In the **Type of database link** list, click **Point/Parameter**.
6. In the **Point** box either type the name of the point or click 🔍 and search for the point using the Point Browser.
7. If you want the alarm state icon to show 'point disabled' or ‘point journal status’ status:

   a. Using the following table, type the name of the appropriate parameter in the **Parameter** box.

For this point type...	...to show point Disabled icon	...to show point Journal Only icon	...to show point Disabled and Journal Only icons
Process (except Foundation Fieldbus)	`almenbstate`	`JournalOnly`	`almenbstate`
Foundation Fieldbus	`alarm_sum.disabled`	`JournalOnly`	`alarm_sum.disabled`
SCADA	`AlarmDisabled`	`JournalOnly`	`AlarmDisabled`
TPS	`alenbst`	Not applicable	`alenbst`

   For more information, see “Understanding alarm state icons for disabled and journal only points.”

   b. Click the **Details** tab and select the **Show point disabled icon**.

8. Configure the icon's other properties as appropriate.

To insert an alarm state icon that is controlled by a script

1. Click ⚠ on the Toolbox toolbar.
2. Move the pointer to the display and drag it diagonally to mark the position of the alarm state icon. (The top-left of the rectangle marks the top-left of the alarm state icon.)
3. Double-click the alarm state icon to open the **Properties Window**.
4. Click the **Data** tab.
5. In the **Type of database link** list, click **None**.
6. Set the icon's default values (the values used before they are changed by the script). For example, if you want the default priority to be low, select **Low**.

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.

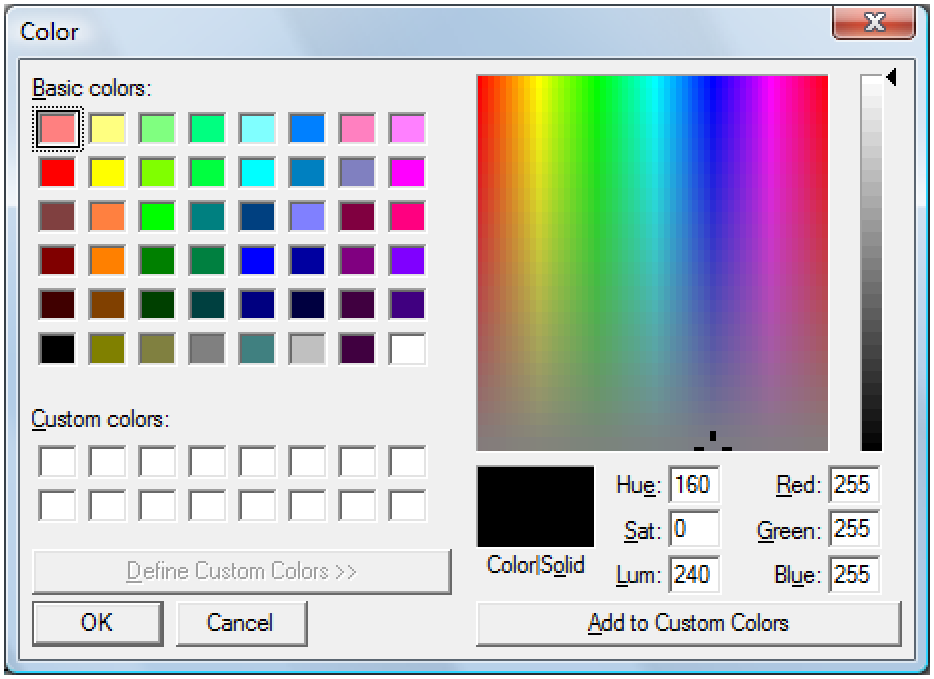
2 If you want to:

- 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.
- 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).

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**.



Hue: 160
Sat: 0
Lum: 240
Red: 255
Green: 255
Blue: 255

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.

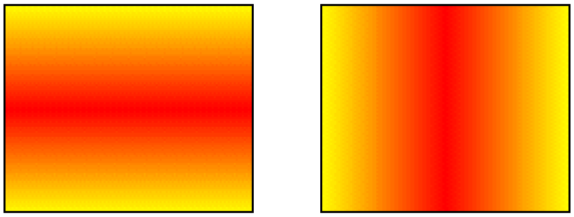


Figure 8: Typical gradient fills—horizontal (left) and vertical (right)

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.)

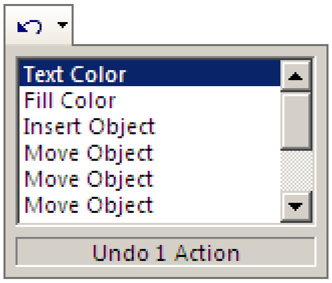
   The Preview box gives an idea of what the gradient will look like.
6. Click **OK**.

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 undo one change in the sequence of changes. If you go back too many steps, click ↷ 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.

Using ActiveX controls in displays

Because of the extremely varied nature of ActiveX controls, we strongly recommend that you thoroughly test the operation of any display that includes ActiveX controls before releasing it to operators.

This section provides guidelines for using ActiveX controls in your displays.

Check that the ActiveX control's execution time is not more than one second

Check that the control does not take more than one second to execute its function or its script. If it takes longer than one second, it must be in its own thread or process (exe).

If you use more than one control, check that the display meets your performance requirements in all conditions.

Check that the ActiveX control does not block operator access to Station

Check that the control never blocks operator access to Station. For example, does not use a modal message box that must be acknowledged before the operator can access the display.

Check that the ActiveX control does not make synchronous calls to the Network API

The control must not make synchronous calls to the Network API because this can result in 'blocking' behavior.

Check that the ActiveX control does not leak memory

Check that the control does not leak memory. Controls run inside the Station process, and the repeated loading of a display on a typical system can eventually exhaust the available resources on the computer. You must therefore thoroughly test the control by repeatedly loading displays containing them, and check the memory and resource usage of *Station.exe* and *HSCStationWindow.exe* in the Windows Task Manager.

Check that you have configured the ActiveX control's specialized properties

Some controls add specialized properties to the Properties Window, which you must configure. (See the documentation/help supplied with the control for details.)

Related topics

"Inserting an ActiveX control" on page 110

Using advanced display features

The following topics describe how to use advanced display features, such as style sheets and shortcut menus.

Related topics

"About style sheets" on page 120
"Attaching a style sheet to a display" on page 121
"About creating a style sheet" on page 123
"About shortcut menus" on page 132
"Creating a custom shortcut menu" on page 133

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

- 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.

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. Before doing this task, consider the following information:

- 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.)

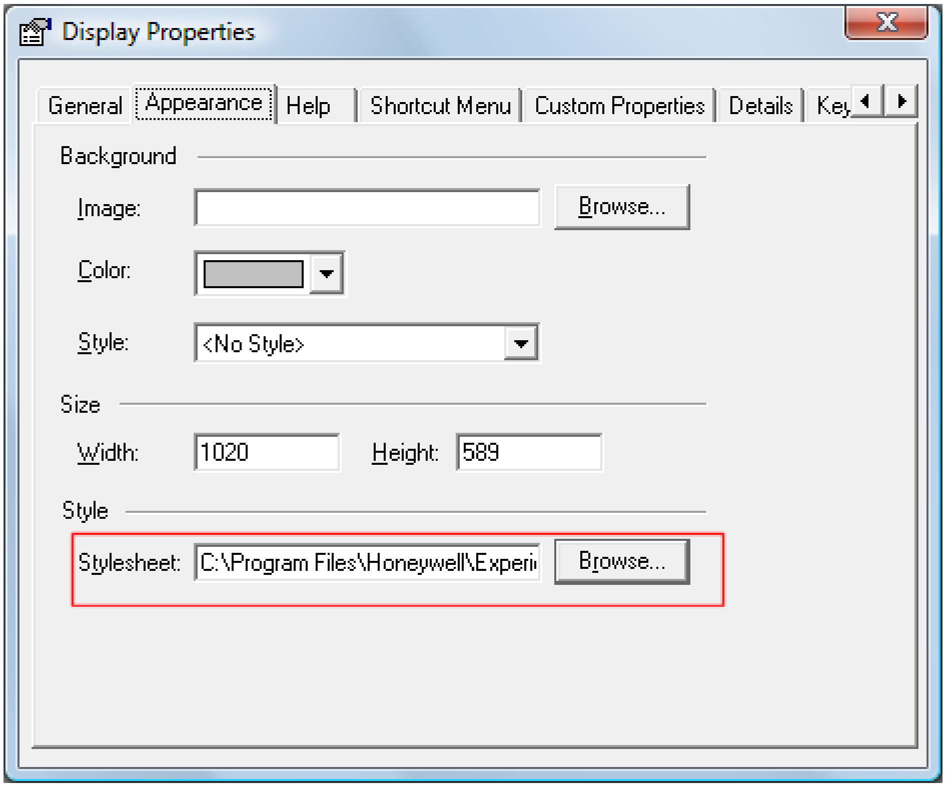


Image:

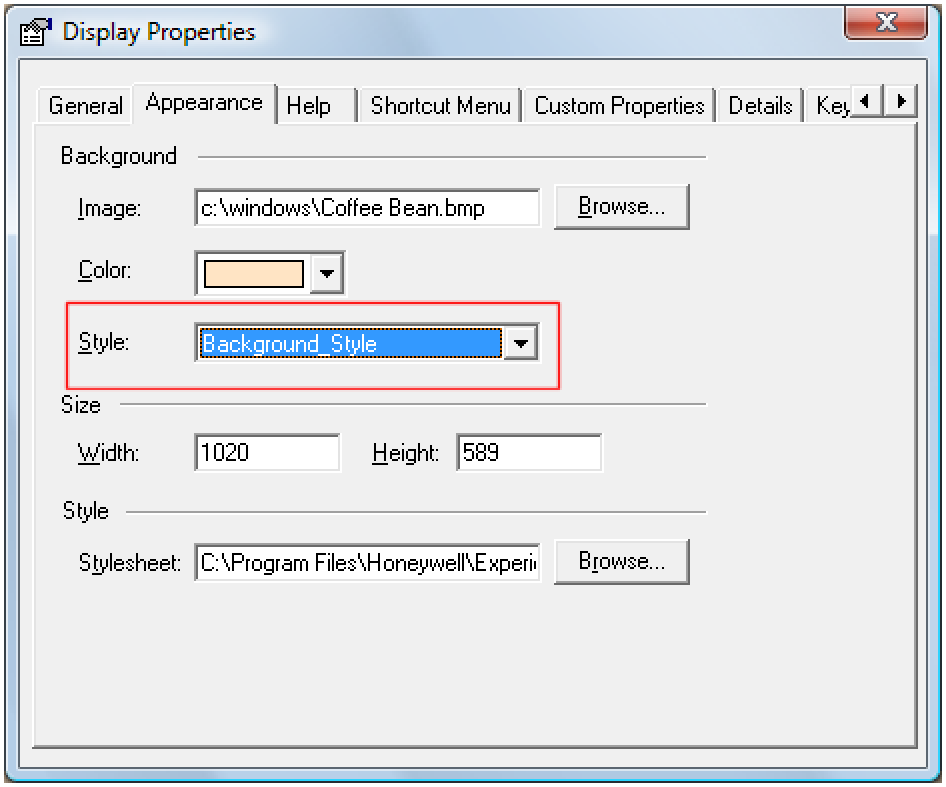
Color:

Style: <No Style>

Width: 1020 Height: 589

Stylesheet: C:\Program Files\Honeywell\Experi

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



Background

Image: c:\windows\Coffee Bean.bmp

Color:

Style: Background_Style

Size

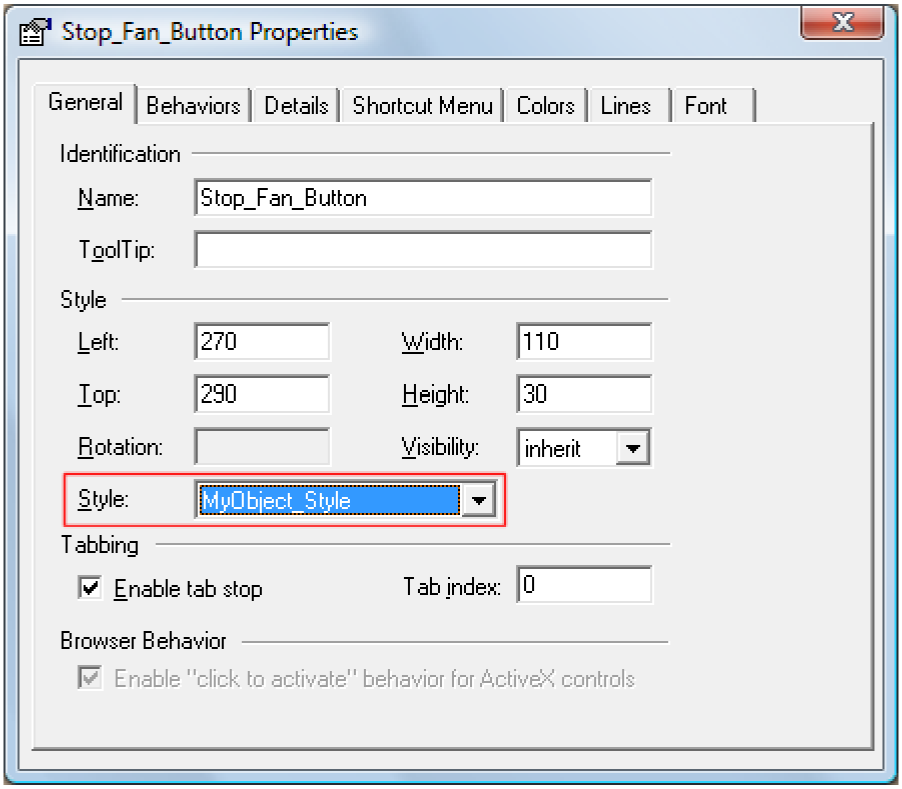
Width: 1020

Height: 589

Style

Stylesheet: C:\Program Files\Honeywell\Experi

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



Identification

Name: Stop_Fan_Button

ToolTip:

Style

Left: 270

Width: 110

Top: 290

Height: 30

Rotation:

Visibility: inherit

Style: MyObject_Style

Tabbing

☑ Enable tab stop

Tab index: 0

Browser Behavior

☑ Enable "click to activate" behavior for ActiveX controls

4 Repeat step 3 as required for the remaining objects.

About 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 *<install folder> \Honeywell\Experion PKS\Client\HMIWeb Display Builder\Examples, where <install folder> is the location where Experion is installed.*

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 reopen 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.

  You can also create your own (user-defined) style attributes.

Syntax of a style attribute definition

Each style attribute definition uses the following syntax:

```
AttributeName : value ;
```

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.

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. For more information about C/C++ string formatting rules, see http://msdn2.microsoft.com/en-us/library/56e442dc(VS.80).aspx.	Alphanumeric
hw-animation-alarm-high-color	Alarm high color.	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).	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.	Page background, alphanumeric, button, checkbox, combobox, vector graphic, indicator, picture, text
hw-element-class	Defines the object types to which a style applies.	All display elements
hw-fill-color	Fill color.	Page background, alphanumeric, button, checkbox, combobox, vector graphic, indicator, text
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

Attribute	Description	Applicable to
hw-hatch-color	Crosshatch line color. Indicates stale quality. When this attribute is defined with a named or hexadecimal color, overrides the default color of *DarkGray* (*#A9A9A9*).	Alphanumeric, combobox, indicator
hw-image-src	The URL or path/name of the image.	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 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.	Shape
hw-text-align	Text alignment: *left* *center* *right* *justify*	Alphanumeric, button, checkbox, combobox, text
hw-text-color	Text color.	Alphanumeric, button, checkbox, combobox, text
hw-text-effect	Text effect: *none* *strikeout* *underline* *italic* *bold* *blink*	Alphanumeric, button, checkbox, combobox, text

Attribute	Description	Applicable to
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

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.
hvg.group.1	Group.
hvg.textbox.1	Textbox.
hvg.combined.1	Combined object.
hsc.alpha.1	Alphanumeric.
hsc.checkbox.1	Checkbox.
hsc.button.1	Pushbutton.
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/en-us/library/aa358802.aspx.
- 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.

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*

Attribute	Equivalent object property
hw-line-effect	*lineColorBlink*
hw-line-end-arrow	*EndArrow*
hw-line-start-arrow	*StartArrow*
hw-line-style	*LineStyle*
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*

Unsupported style properties

The following standard CSS style properties are not supported.

- text-align
- text-shadow
- letter-spacing
- word-spacing
- white-space

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

- Display elements can be assigned a base style class. In this example, VectorAndAlpha_Style is a base style class.
- 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 its 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 on a display object or the display.

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

There are also default display-level shortcut menus that are used if the user right-clicks on:

- The display.
- A display object that is not bound to a point parameter.

Notes

- You use **Shortcut** properties to assign a custom shortcut menu to a display or a display 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 shape sequence.

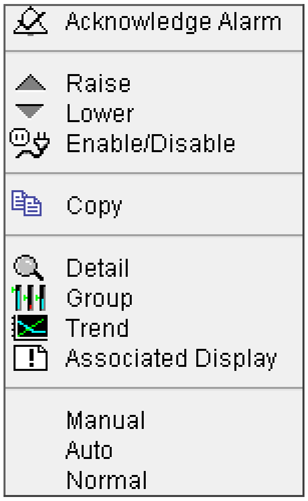


Figure 9: Example shortcut menu (for an analog point)

Related topics

"About copy and paste and drag and drop behaviors" on page 208

"Shortcut properties" on page 268

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 file, save it with a new name and edit it as required.

- The standard shortcut menus are stored in *<data folder>\Honeywell\Experion PKS\Client\Abstract\*, where *<data folder>* is the location where Experion data is stored. For default installations, *<data folder>* is *C:\ProgramData*.
- 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 '&' (ampersand) or '>' (greater than symbol) in a menu item, you must use its HTML *named entity* or Unicode value. For example, if you want a menu item called 'Reports & Trends' containing an ampersand, you would use the named entity '&amp;' or the Unicode value '&#0026;' for the ampersand. For a listing of HTML named entities, see the MSDN topic **MSDN** > **MSDN Library** > **Web Development** > **Internet Explorer Development** > **Globalization** > **HTML Character Sets** > **Additional Named Entities for HTML** (see http://msdn2.microsoft.com/en-us/library/aa752008(VS.85).aspx).

Tip

You can also view the Unicode values for a character using the charmap utility on Windows. Use the Unicode value in the format '&#*nnnn*;' in place of the named entity.

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.

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.

`<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.

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

`<html>`
.
.
`</html>`
	HTML code that is not parsed by the XML translator, but is rendered directly in the shortcut menu. (When a shortcut 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.
	• 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.

`<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 *<data folder>\Honeywell\Experion PKS\Client\MenusAndToolbars\Toolbar\* folder. Where *<data folder>* is the location where Experion data is stored. For default installations, *<data folder>* is *C:\ProgramData*. The *C:\ProgramData* folder is a system folder, which means that it is only visible if you select the **Show hidden files, folders, and drives** option button in the **Folder Options** dialog box. To change this setting in Windows Explorer, click **Organize** > **Folder and search options**, and then click the **View** tab.

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.

Property	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.
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 after calling InvokeShortcutMenu:

window.event.cancelBubble=true

Otherwise the default system shortcut menu will appear overriding the menu brought up via a script.

Configuring a 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.customShortcutMenuSourceElement;
  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>
  <mage>.\Images\image1.png</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.png</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.png</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.

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.png</image>
</menuitem>
```

If the shortcut menu is not associated with the current object (that is, the object that has focus), you use *parentWindow.document.customShortcutMenuSourceElement* 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.customShortcutMenuSourceEl
ement)){
      this.disabled=true;
   }
     return true;
  </script>
  <text>Raise</text>
  <image>.\Toolbar\tb012.png</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.customShortcutMenuSourceElement.id == "shape001")&&
(window.external.parentWindow.document.protocol == "http")){
      this.disabled=true;
    }
      return true;
  </script>
  <text>Raise</text>
  <image>.\Toolbar\tb012.png</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.customShortcutMenuSourceElement.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 custom shortcut menu with the InvokeShortcutMenu method.

The following example invokes a shortcut menu, which appears at the current cursor location. (Note that the menu should be stored in the same folder as the current display page.)

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

Optimizing display performance

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

Related topics

“Using the Display Performance Analysis tool” on page 140
“What is the Display Performance Analysis tool?” on page 141
“Display-creation guidelines” on page 144
“Station-related guidelines” on page 151
“Example: Improving the performance of an existing display” on page 152

Using the Display Performance Analysis tool

You can use the Display Performance Analysis tool to check a display's overall size and *performance indicators*.

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.

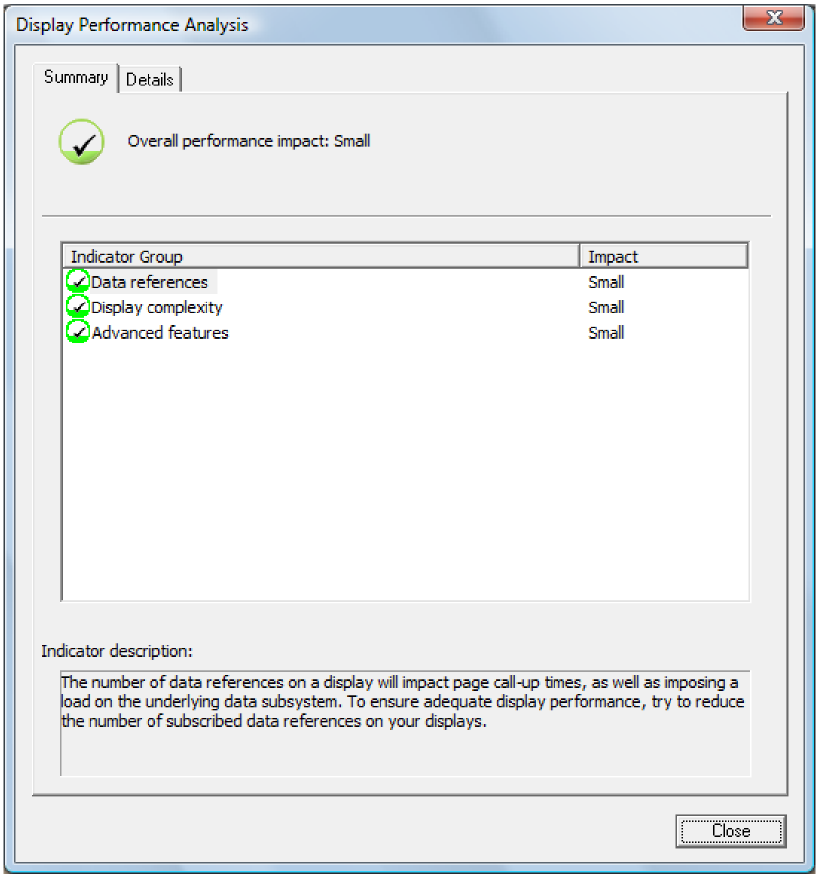
Related topics

“Object count limits” on page 64

What is the Display Performance Analysis tool?

The Display Performance Analysis tool analyzes a display to determine its 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 figure 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.



✓ Overall performance impact: Small

Indicator Group	Impact
✓ Data references	Small
✓ Display complexity	Small
✓ Advanced features	Small

Indicator description:

The number of data references on a display will impact page call-up times, as well as imposing a load on the underlying data subsystem. To ensure adequate display performance, try to reduce the number of subscribed data references on your displays.

Figure 10: 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.

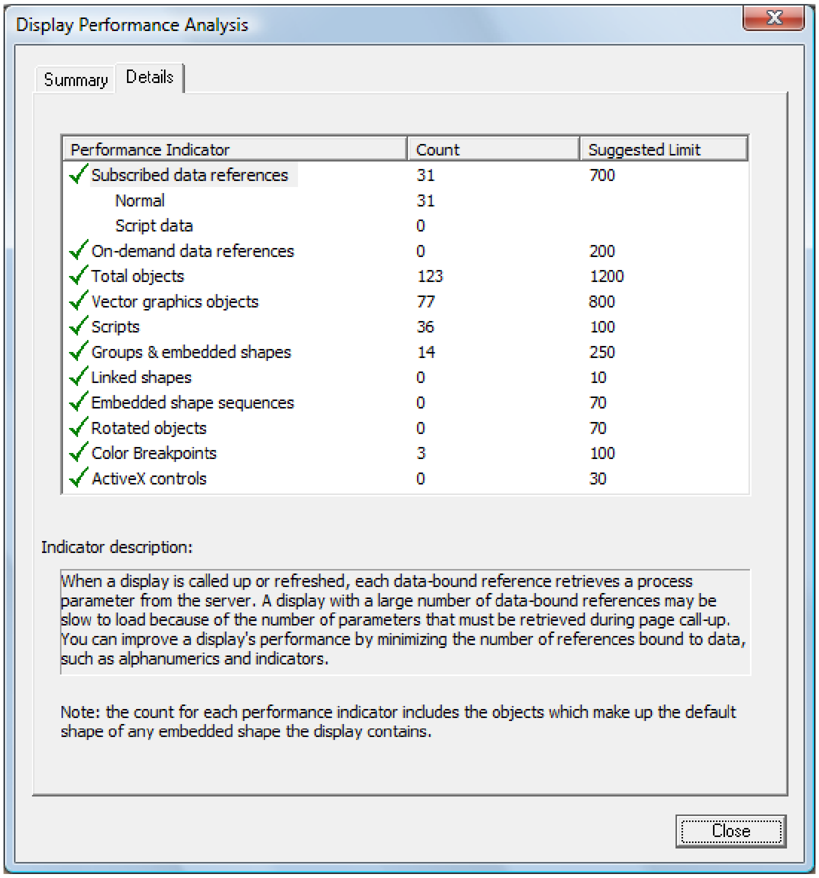
- DSA data

  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.



Performance Indicator	Count	Suggested Limit
✓ Subscribed data references	31	700
Normal	31	
Script data	0	
✓ On-demand data references	0	200
✓ Total objects	123	1200
✓ Vector graphics objects	77	800
✓ Scripts	36	100
✓ Groups & embedded shapes	14	250
✓ Linked shapes	0	10
✓ Embedded shape sequences	0	70
✓ Rotated objects	0	70
✓ Color Breakpoints	3	100
✓ ActiveX controls	0	30

Indicator description:

When a display is called up or refreshed, each data-bound reference retrieves a process parameter from the server. A display with a large number of data-bound references may be slow to load because of the number of parameters that must be retrieved during page call-up. You can improve a display's performance by minimizing the number of references bound to data, such as alphanumerics and indicators.

Note: the count for each performance indicator includes the objects which make up the default shape of any embedded shape the display contains.

Figure 11: 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.

Display-creation guidelines

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

Related topics

"Display resolution" on page 144
"Guidelines for pan and zoom displays" on page 145
"Minimizing the object count" on page 146
"Using shapes" on page 147
"Creating your own shapes" on page 147
"Minimizing the use of behaviors" on page 148
"Using the basic trend object instead of the trend object" on page 148
"Using scripts" on page 148
"Minimizing the number of scripts" on page 148
"Referencing objects" on page 149
"Errors in Scripts" on page 149
"On demand script data" on page 149
"Using InvokeTask and RequestServerLRN functions in scripts" on page 149
"Working with User Account Control" on page 149
"Minimizing the use of ActiveX controls" on page 150
"Configuring Display Properties" on page 150

Display resolution

Tip

Experion uses the default zoom level of **Zoom To Fit**, which displays the contents of the Station window without the use of vertical and horizontal scroll bars. Therefore, display resolution considerations are necessary only when operators use your displays at other magnification levels.

When creating displays in HMIWeb Display Builder, consider the maximum screen real estate available to you (number of 'pixels') and plan your design accordingly. (Pixel is short for 'picture element.' One pixel is a measurement representing the smallest amount of information displayed graphically on the screen as a single dot.) Other elements on the screen, such as the Windows Taskbar and the Station window, will use some of the available pixels, so you need to subtract those pixels from your design.

For example, if the screen resolution that operators use to view your custom display is 1600×1200 pixels, and the other elements on the screen use 50×200 pixels, then the maximum screen real estate available to you is 1550×1000 pixels.

Consider the following elements that reduce the amount of available screen real estate.

Station window

The Station window in which your display can appear comprises of the following elements:

- Title bar (18 pixels)
- Menu bar (24 pixels)
- Toolbar, and Command zone (24 pixels)
- Message zone (24 pixels)
- Location pane, and Display (in which your display can appear) (variable number of pixels)
- Alarm line (18 pixels)
- Status Bar (24 pixels)

Therefore, the Station window uses approximately 132 pixels of vertical space that is reserved elements other than your display.

In addition, the Station window uses approximately 26 pixels of horizontal space that is reserved elements other than your display.

Tabbed displays

If tabbed displays are enabled in your system, you need to allow an additional 13 pixels of vertical space for the tabs to appear without causing Station to draw vertical scroll bars.

Attention

Tabbed displays are not supported within eServer.

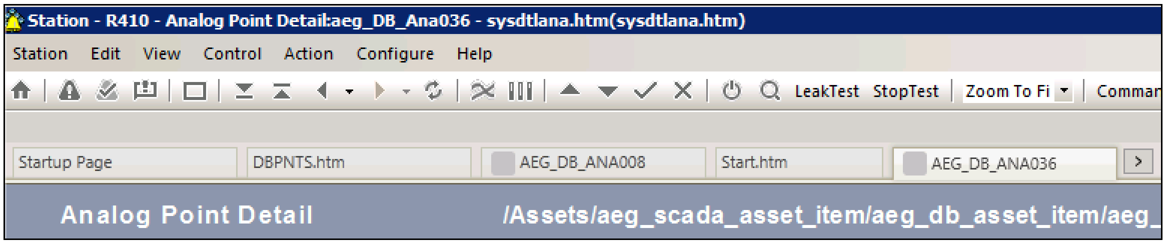


Figure 12: An example of tabbed displays in single window Station

Windows Taskbar

The standard height of the Windows Taskbar is 22 pixels. However, when you open more programs than will fit on the Taskbar, scroll bars appear and increase the size to 24 pixels.

Adding it all up

If you add all of the vertical and horizontal space used by other elements on the screen, here is how it looks:

Element	Pixels used
Station window	26×132 pixels
Tabbed displays (optional)	0×13 pixels
Windows Taskbar	0×22 pixels
Total	26×167 pixels

So, given the example above where you started out with 1600×1200 pixels, you need to subtract 167×26 pixels to arrive at the maximum size of your display (i.e., 1574×1033 pixels).

Guidelines for pan and zoom displays

Pan and Zoom displays are much like normal displays, but they have the potential to contain a lot more data than a normal display, meaning that a few guidelines should be followed to ensure they perform as expected:

- Don’t overload pan and zoom displays with content. Even though display elements outside the viewport do not subscribe to data, the html content is still loaded. The performance archived when the page loads and when panning or zooming depends on both the amount of content currently within the viewport and the overall size of the display.
- Ensure that zooming is disabled for the pan and zoom display. Like other displays, zooming is disabled by default, but can be enabled in the **Page Details** section of the **Properties Grid**. It is recommended that you do not do this for very large displays, as it will cause Station to try to zoom to fit when loading the display, potentially subscribing to a large amount of data.

- If using a background image on the display:
  - Use an appropriate file format, such as *jpeg* or *png*, with compression to reduce the image size. If a large color palette is not needed consider using an 8bit png (256 colors)
  - Resize the image to match the display bounds so the image does not need scaling when rendered
  - If the image dimensions are the same as the display dimensions, consider setting the image as the background of the display rather than adding the image to the display. This will further reduce scaling.
- Design the display with the target monitor resolution in mind. A Pan and Zoom display effectively renders at 100% zoom so invoking the same display on a larger resolution monitor will result in more load on the control network as a larger number of data references are visible.
  - Try to reduce the density of data references on the display so that the maximum number of data references visible at any one time is as few as possible. Use the display performance analysis numbers as a guideline.
- Use existing techniques such as combining HVG objects to images to reduce the overall object count of the Pan and Zoom display.

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.
- 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.
- 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.

Using shapes

You should use the shapes supplied with HMIWeb Display Builder wherever possible because they have been optimized for display performance.

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.

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
```

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 the basic trend object instead of the trend object

HMIWeb supports two types of trend object:

- **Trend**, which includes a rich set of features and allows you to give operators extensive control over how data is displayed in the trend.
- **Basic trend**, which includes only basic features and gives operators limited control over the trend.

Due to the complexity of the trend object, Honeywell strongly recommends that you only include one instance of it in a display. (Including more than one instance may degrade the display's performance.) Use the basic trend object if you want to include several trends on a display.

Using scripts

This section includes guidelines you need to consider when using scripts.

To learn about:
“Minimizing the number of scripts” on page 148
“Referencing objects” on page 149
“Errors in Scripts” on page 149
“On demand script data” on page 149
“Using InvokeTask and RequestServerLRN functions in scripts” on page 149

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.

In the situation where several display elements co-exist on the same display, each with its own element scripting, such as with animation, it is more efficient to create and utilize a common event handler, using a page level boolean variable to flag when to start the page animation scripting.

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

Although you can use the VBScript 'on error resume next' statement to hide errors in scripts, you should avoid its use as far as possible. If you do use it, you should only do so after you have tested your script as extensively as possible.

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.

Attention

You cannot write to on demand references because they are read-only.

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.

Working with User Account Control

The new security features in the Windows operating systems include elevation prompts if a task is performed that requires administrator privileges. For example, if an ActiveX control on a display initiates the installation of components, the operator is prompted for a user name and password with the appropriate privileges. This is known as an elevation prompt.

To prevent the loss of view of the Experion system, you should not add any components to your custom displays such as scripts or ActiveX controls that perform actions requiring administrator privileges.

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.

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 'Connection Properties' in the '*Customizing Stations*' section of the *Configuration Guide*.

Example: Improving the performance of an existing display

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

The example 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).

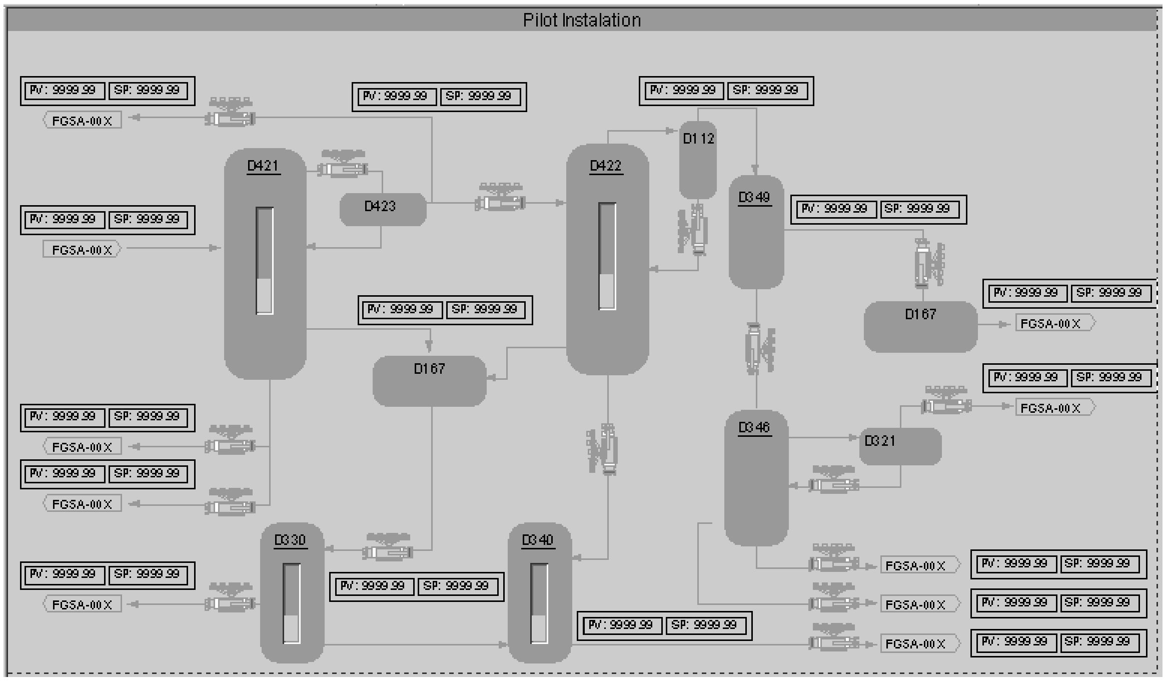
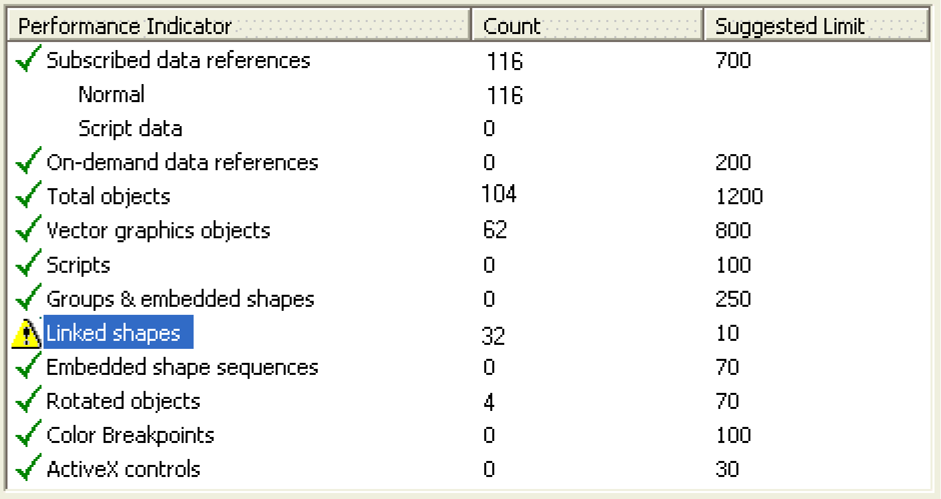


Figure 13: The display

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.



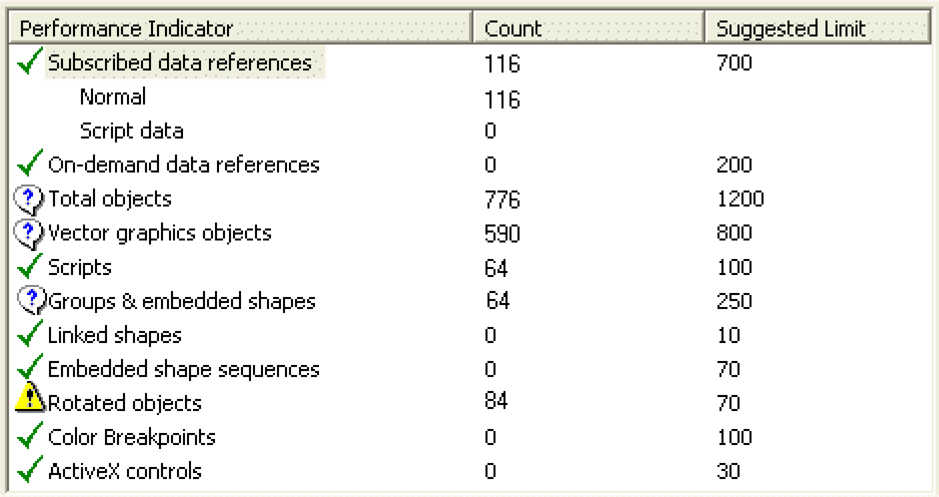
Performance Indicator	Count	Suggested Limit
✓ Subscribed data references	116	700
Normal	116	
Script data	0	
✓ On-demand data references	0	200
✓ Total objects	104	1200
✓ Vector graphics objects	62	800
✓ Scripts	0	100
✓ Groups & embedded shapes	0	250
⚠ Linked shapes	32	10
✓ Embedded shape sequences	0	70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

Figure 14: The performance indicators for the original display

2 Because the number of linked shapes exceeds the recommended limit, you embed them.

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.)



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

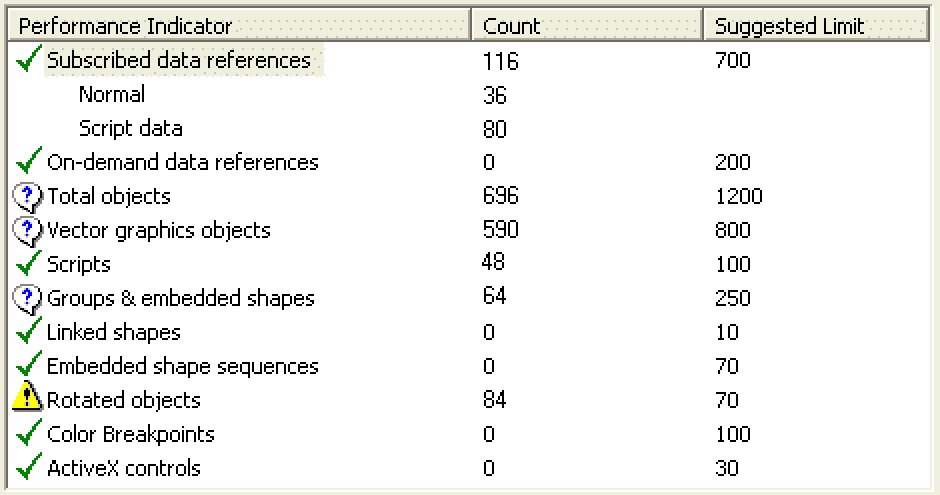
Figure 15: The performance indicators after embedding the shapes

4 Because there are a large number of scripts and data-bound objects (dynamic objects), which you know belong to a particular dynamic shape, you decide to improve the shape's design.

5 Upgrade the embedded shapes that are based on this shape.

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.



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

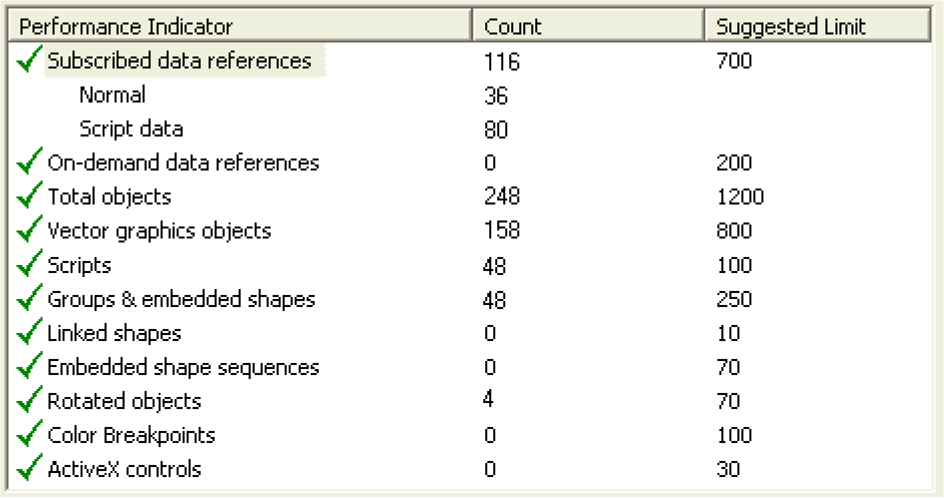
Figure 16: The performance indicators after redesigning the dynamic shape

7 Because you know that the static shape (the valve) includes many objects, you decide to combine its objects to create a single bitmap.

8 Upgrade the embedded shapes that are based on this shape.

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.



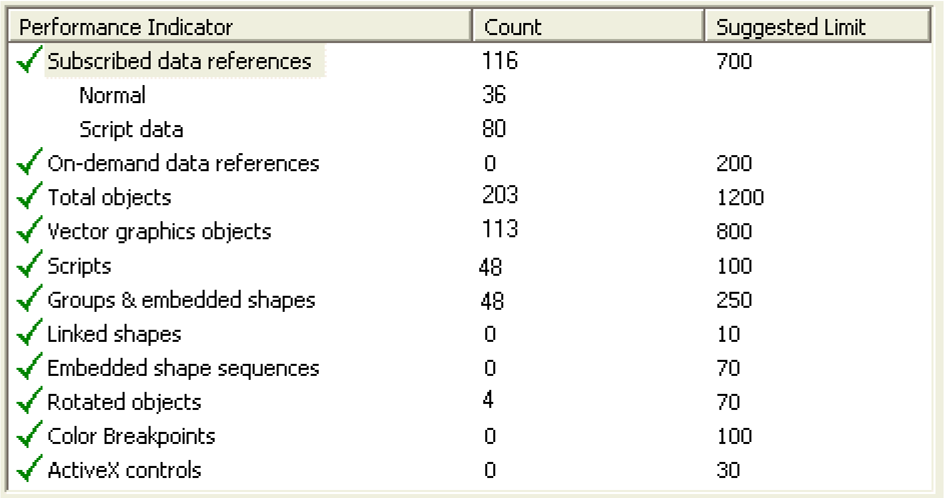
Performance Indicator	Count	Suggested Limit
✓ Subscribed data references	116	700
Normal	36	
Script data	80	
✓ On-demand data references	0	200
✓ Total objects	248	1200
✓ Vector graphics objects	158	800
✓ Scripts	48	100
✓ Groups & embedded shapes	48	250
✓ Linked shapes	0	10
✓ Embedded shape sequences	0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

Figure 17: The performance indicators after redesigning the static shape

10 Because the display's background includes many vector graphic objects, you decide to combine them to create a single bitmap.

11 Use the Display Performance Analysis tool again.

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



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

Figure 18: The performance indicators after redesigning the static shape

Example: Improving the design of a dynamic shape

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

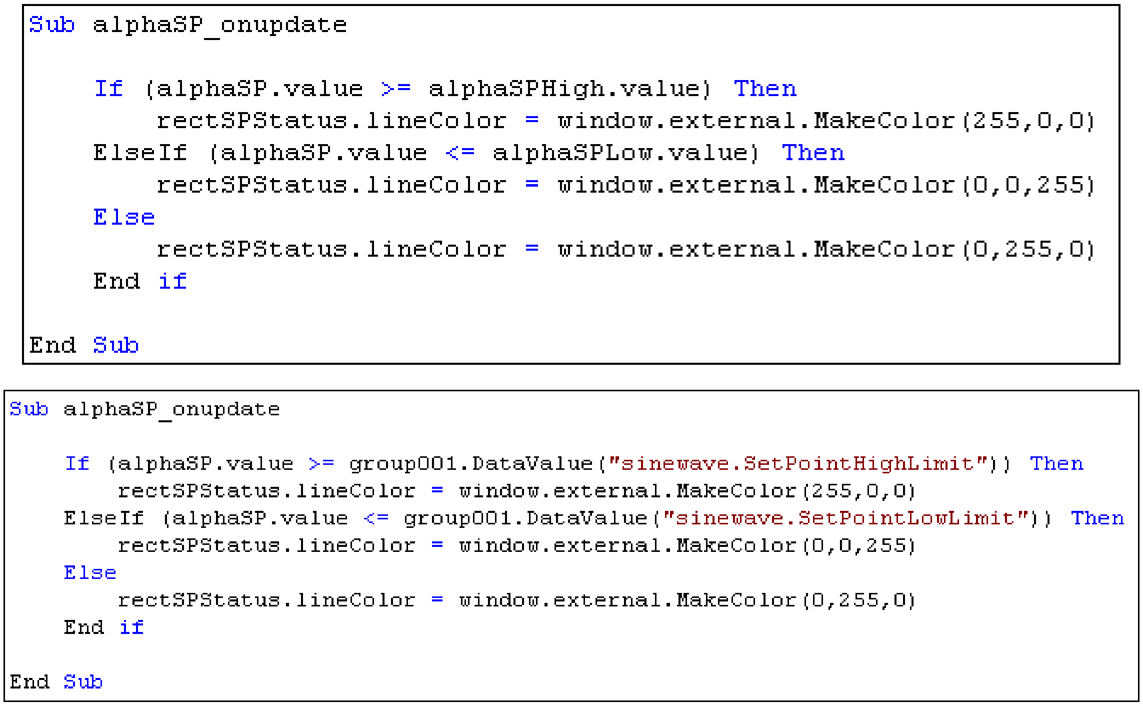
The recommended way of accessing parameter values is to add them to the **Script Data** tab of a suitable object. (This means that you no longer need the five invisible alphanumerics.)

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.)

To improve the shape's performance

1 Add the required point parameters to the Script Data tab of the group object.

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.



```
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
```

Figure 19: A typical script (original above, updated below)

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.

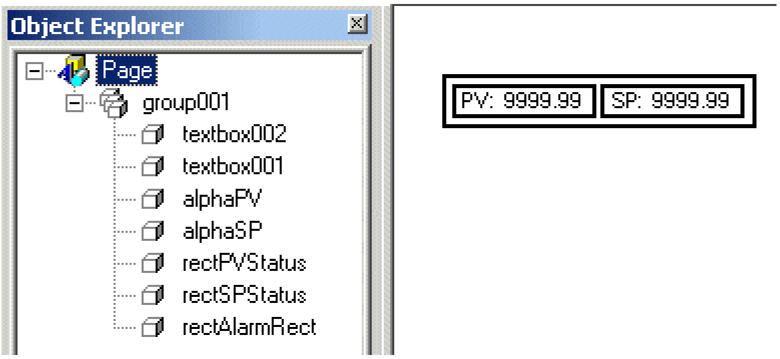


Figure 20: The updated shape, with its objects shown in the Object Explorer

Optimizing the performance of shapes containing similar scripts

If you have many shapes on a display that contain a similar body of script repeated for each shape, you should amalgamate the scripts to a common place, then call the common script from the shape event, passing in the shape name for identification and action. This minimizes the amount of duplicate scripting, thus optimizing 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 CDA_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.

Using the diagnostic and productivity tools

The following topics describe how to use the diagnostic and productivity tools in HMIWeb Display Builder.

Related topics

Using the HMIWeb Display Builder Assistant

You can use the HMIWeb Display Builder Assistant to perform the following tasks on a set of displays and shapes.

Task	Description
Rename Points	Searches for, and changes, the name of a point in each display and shape file in the specified folder, and subfolders if selected.
Embed Shapes	Embeds the linked shapes in each display and shape in the specified folder, and subfolders if selected.
Upgrade Embedded Shapes	Upgrades any embedded shapes in each display and shape in the specified folder, and subfolders if selected.
Replace Shapes	Replaces shapes across multiple displays and shape files in the specified folder, and subfolders if selected.
Archive Displays & Shapes	Creates an archive file for each display and shape file in the specified folder, and subfolders if selected.
Unarchive Displays & Shapes	Creates a display/shape for each archive file in the specified folder, and subfolders if selected.
Create Display Report	Creates a report on the displays in the specified folder, and subfolders if selected. You specify what you want to include in the report. For example, a report can include an analysis of the performance, scripts and custom properties.
Performance Analyze	Creates a performance analysis report of the displays in the specified folder, and subfolders if selected. If Microsoft Excel is installed, the performance analysis report is generated as an Excel spreadsheet file. Otherwise, if Microsoft Excel is not installed, the performance analysis report is generated in the CSV format and the file name is *DisplayPerformanceReport.csv*. Select the folder where the report is to be saved.
Validate Displays	Creates a display validation report of the displays in the specified folder, and subfolders if selected. You can select whether to perform an online or offline validation. If Microsoft Excel is installed, the validate displays report is generated as an Excel spreadsheet file. Otherwise, if Microsoft Excel is not installed, the validate displays report is generated in the CSV format and the file name is *DisplayAnalysisReport.csv*. Select the folder where the report is to be saved.
Generate Alarm Groups	Creates alarm group definition files for each display in the specified folder, and subfolders if selected. You can also generate a single alarm group definition file that generates a single alarm group for all of the displays in the specified folder.

To perform a task

1. Choose **Tools** > **Display Builder Assistant**.
   The **HMIWeb Display Builder Assistant** dialog appears.
2. Select the task you want to perform, enter any relevant details, and then click **Run**.

Changing the server

To change the server

1. Choose **Tools** > **Change Server**.
   The **Change Server** dialog appears.
2. Select the server you want from the **Default Acronym and Point Browser Server** list, and then click **OK**.

Generating a display report

You use the Display Report tool to create a report about a display. Using the HMIWeb Display Builder Assistant tool, you can create a report about all displays located in a specific folder.

When you request a report, you specify what you want to include in the report. For example, a report can include an analysis of the performance, scripts and custom properties.

To generate a display report for a single display

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
Performance analysis	This includes whether the display passes a performance analysis. This analysis also includes the recommended limits and total counts within the display for all objects, vector graphic objects, linked shapes, embedded shape sequences, groups and embedded shapes, rotated objects, color breakpoints, subscribed data objects, script data, on demand data objects, and ActiveX controls.
Object properties	This includes name, style class, behaviors, bound data, display format, script data, animation details, and hierarchy.
Point information	This includes information about points bound to display objects, such as update rates.
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.

To generate a display report for all displays contained in a specified folder

1. Choose **Tools** > **Display Builder Assistant**.

   The HMIWeb Display Builder Assistant appears.
2. In the **Task** list, click **Create Display Report**.
3. In the **Display folder** box, type the path to folder containing the displays to include in the report, or click **Browse** to select this folder name.
4. If you want to include displays contained in the subfolders of the folder displayed in the **Display folder** box, select **Include subfolders**.
5. In the **Save report to** box, type the path to the folder where you want to save the report, or click **Browse** to select the folder name.
6. In the **Report name** box, type the report name.

7 In the **Report file format** list, select the format of the report.

You can save the report as an HTML page or an XML file.

8 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
Performance analysis	This includes whether the display passes a performance analysis. This analysis also includes the recommended limits and total counts within the display for all objects, vector graphic objects, linked shapes, embedded shape sequences, groups and embedded shapes, rotated objects, color breakpoints, subscribed data objects, script data, on demand data objects, and ActiveX controls.
Object properties	This includes name, style class, behaviors, bound data, display format, script data, animation details, and hierarchy.
Point information	This includes information about points bound to display objects, such as update rates.
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.

9 Click **Run**.

The report is saved in the location you specified. Generating the report may take several minutes.

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.

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 display 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.

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**.

Viewing a display in Station

To ensure that a display you are editing has been configured correctly, you can view it in Station with live data before it is deployed to the actual location where it will be accessed.

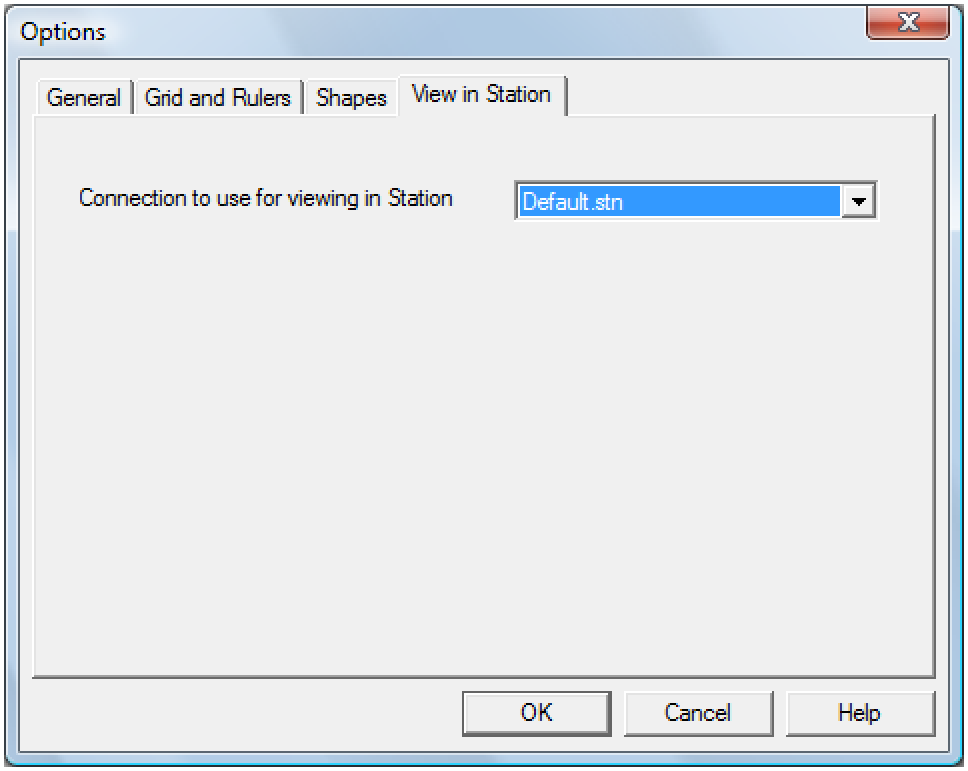
You can copy a display to a temporary location, edit the display in HMIWeb Display Builder and test the modified display using **View in Station** before deploying it to a location where it can be accessed by other stations.

To configure the settings related to **View in Station** you need to set up the connection that Station will use to preview the display.

Selecting the connection that station must use to preview the display

1 Choose **Tools** > **Options**.

2 Click the **View in Station** tab.



Connection to use for viewing in Station: Default.stn

3 Select the connection you would like the station to use to connect to the server required for viewing your display with live data.

4 Click **OK**.

> **Attention**
> If the display being edited is a popup or faceplate, it must be embedded in another display before you can view it in Station.

Viewing your display in Station

1 Open the display you need to preview in Station.

2 Either,

- Choose **File** > **View in Station**.
- Click ! on the Standard toolbar.

The display will launch in Station.

Validating displays

Before deploying a display, you should validate the display to ensure that it contains no configuration errors.

> **Attention**
> If you validate a display from a Console Station, HMIWeb Display Builder must be connected to the Experion server, not the Console.

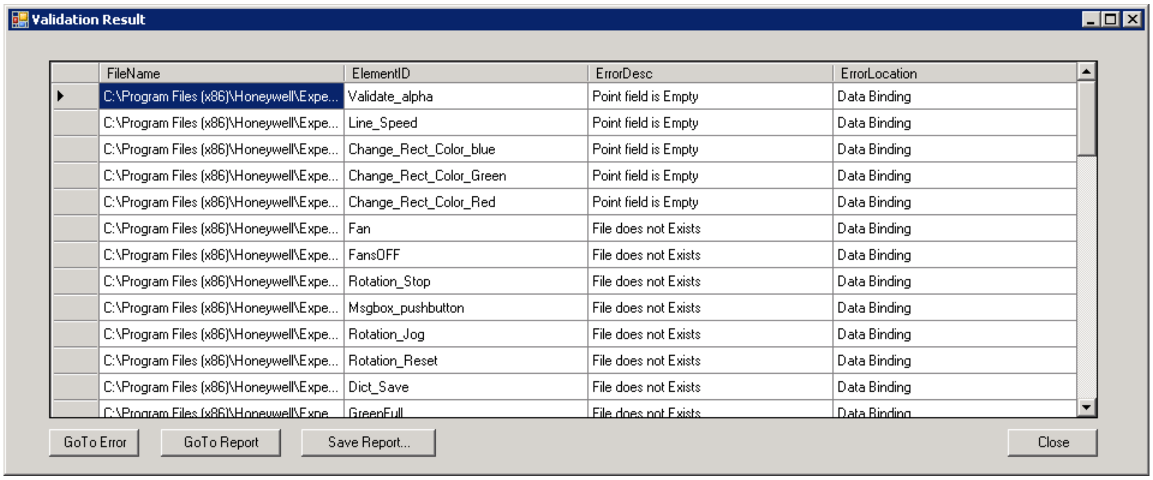
To validate a display

1 Choose **File** > **Validate**.
The **Validate Display** dialog box appears.

2 Choose the validation type to perform:

Option	Description
Offline	You can validate the display without being connected to the Experion server. Checks for missing display data, and missing references to style sheets, menus, and files.
Online	You must be connected to the Experion server. Performs offline validation checks (missing display data, and missing references to style sheets, menus, and files), as well as validating display references to point/parameters.

3 Click **Validate**.
The display is validated. If there are validation errors, the **Validation Result** dialog box appears



FileName	ElementID	ErrorDesc	ErrorLocation
C:\Program Files (x86)\Honeywell\Expe...	Validate_alpha	Point field is Empty	Data Binding
C:\Program Files (x86)\Honeywell\Expe...	Line_Speed	Point field is Empty	Data Binding
C:\Program Files (x86)\Honeywell\Expe...	Change_Rect_Color_blue	Point field is Empty	Data Binding
C:\Program Files (x86)\Honeywell\Expe...	Change_Rect_Color_Green	Point field is Empty	Data Binding
C:\Program Files (x86)\Honeywell\Expe...	Change_Rect_Color_Red	Point field is Empty	Data Binding
C:\Program Files (x86)\Honeywell\Expe...	Fan	File does not Exists	Data Binding
C:\Program Files (x86)\Honeywell\Expe...	FansOFF	File does not Exists	Data Binding
C:\Program Files (x86)\Honeywell\Expe...	Rotation_Stop	File does not Exists	Data Binding
C:\Program Files (x86)\Honeywell\Expe...	Msgbox_pushbutton	File does not Exists	Data Binding
C:\Program Files (x86)\Honeywell\Expe...	Rotation_Jog	File does not Exists	Data Binding
C:\Program Files (x86)\Honeywell\Expe...	Rotation_Reset	File does not Exists	Data Binding
C:\Program Files (x86)\Honeywell\Expe...	Dict_Save	File does not Exists	Data Binding
C:\Program Files (x86)\Honeywell\Expe...	GreenFull	File does not Exists	Data Binding

Figure 21: Example Validation Result dialog box

Each error is displayed as a row in the **Validation Result** dialog box.

Column	Description
FileName	The folder location of the display that contains the validation error.
ElementID	The object name that contains the validation error.
ErrorDesc	A short description of the validation error.

Column	Description
ErrorLocation	The location within the display or display object of the validation error. Common error locations are: • Data Binding - the error is within the data properties of the display object. • File Reference Binding - the error is due to either invalid or missing file references. • Display Custom Property - the error is within the display's custom properties. • Script - the error is within a display or display object script.

4 If the **Validation Result** dialog box appears, correct any display validation errors.

The **Validation Result** dialog box displays each of the errors. For more information about each error:

a In the **Validation Result** dialog box, select the error line.

b Click **Go To Error**.

The validation errors include:

Validation Error	Description
Scripting errors	When you click **Go To Error**, HMIWeb Display Builder opens the Script Editor and highlights the script line that contains the error. In the Script Debugger application, the location of the error is highlighted in blue.
Embedded shape errors	When you click **Go To Error**, HMIWeb Display Builder opens the shape properties for the shape display object. When you click **Go To Error** and the "Unable to reach the error location. Validate the original shape file to see the error location." message appears, you will need to open the shape file of the embedded shape and then validate this file to locate the error.
Display object errors	When you click **Go To Error**, HMIWeb Display Builder opens the display properties and highlights the object property value containing the error.
Performance impact errors	When you click **Go To Error**, HMIWeb Display Builder opens the **Display Performance Analysis** window and lists the performance indicators related to this display.

5 If you want to export the validation result list, click one of the following:

Option	Description
Go To Report	The validation report is exported as a CSV file, to the *DisplayAnalysisReport.csv* file. If you have an application associated with this file type, the report appears in this application.
Save Report	The validation report is exported as a CSV file, which you can save to a folder.

6 Click **Close** to close the **Validation Error List** dialog box.

Differences between online validation and offline validation

Offline validation	Online validation
Offline validation checks for following types of errors: • Display elements that have missing references in embedded shapes and within the display. • Script Data objects or Custom properties that have missing references in embedded shapes and within the display. • Invalid popup, shortcut, and stylesheet file references in embedded shapes and within the display. • Invalid references to a style definitions that do not exist in the stylesheet file. • Checks if the performance parameters of the display are within the suggested limits. Offline validation can be performed without connecting to the Experion server.	Online validation checks for the same type of errors as offline validation, plus these other error types: • Display elements that have invalid references in embedded shapes and within the display. • Script Data objects or Custom properties that have invalid references in embedded shapes and within the display. Online validation can only be performed when connected to the Experion server.

Display and display objects

Related topics

ActiveX Control display object

Description

Creates a link to an ActiveX Control.

Properties

- “ActiveX Properties properties” on page 217
- “Behaviors properties” on page 224
- “General (Display object) properties” on page 249
- “Shortcut properties” on page 268

ActiveX Document display object

Description

Create a link to an ActiveX document, such as a Word document, with a display.

Properties

- “Behaviors properties” on page 224
- “General (Display object) properties” on page 249
- “Shortcut properties” on page 268

Alarm State display object

Description

Creates an alarm state icon (as used in the Alarm Summary) that displays an alarm state.

Properties

- “Behaviors properties” on page 224
- “Colors properties” on page 228
- “Data properties” on page 232
- “Details (Alarm State) properties” on page 237
- “General (Display object) properties” on page 249
- “Shortcut properties” on page 268

Alarm Table display object

Description

Creates a specialized table that lists alarms.

Properties

- “Colors properties” on page 228
- “Columns properties” on page 229
- “Details (Alarm Table and Event Table) properties” on page 239
- “Font properties” on page 248
- “General (Display object) properties” on page 249
- “Lines properties” on page 254

Related topics

“Alert Table display object” on page 179
“Event Table display object” on page 188
“Message Table display object” on page 193
“About alarm, alert, event, and message tables” on page 175

About alarm, alert, event, and message tables

Alarm, alert, and message tables

These objects display information from the same source as their counterpart Summary window in Station.

- An *alarm table* lists alarms in a manner similar to the Alarm Summary.
- An *alert table* lists alerts in a manner similar to the Alert Summary.
- A *message table* lists messages in a manner similar to the Message Summary.

The properties you can specify when creating an alarm, alert, or message table include:

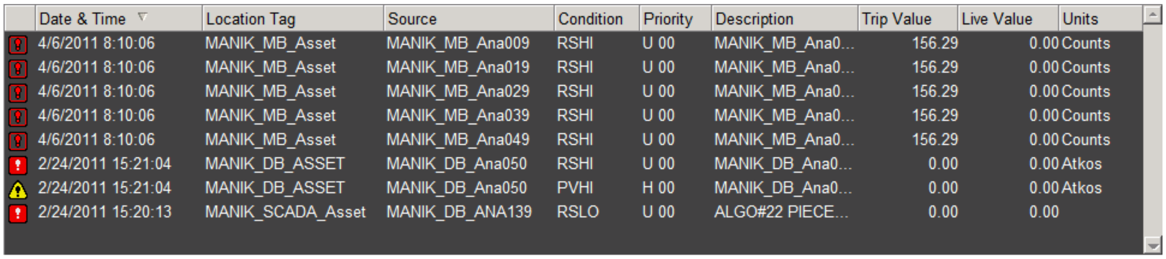
- The types of alarms, alerts, and messages that are shown. (For example, you may only want to show urgent alarms for a particular location.)
- The details that are shown (date and time, description, priority, and so on), and the order in which they appear across the table.
- Whether there are horizontal and vertical scroll bars.

Attention

Alarm and alert tables on custom displays do not show shelved or suppressed notifications.

To see an alarm table that includes shelved alarms (or alerts), you need to call up the Alarm Summary (or Alert Summary) and choose the **(shelved alarms)** (or **shelved alerts**) view or choose the **Shelved** filter for the alarm (or alert) icon column.

To see an alarm table that includes suppressed alarms, you need to call up the Alarm Summary, select the **(suppressed alarms)** view or choose the **Suppressed** filter for the alarm icon column.



	Date & Time ▽	Location Tag	Source	Condition	Priority	Description	Trip Value	Live Value	Units
!	4/6/2011 8:10:06	MANIK_MB_Asset	MANIK_MB_Ana009	RSHI	U 00	MANIK_MB_Ana0...	156.29	0.00	Counts
!	4/6/2011 8:10:06	MANIK_MB_Asset	MANIK_MB_Ana019	RSHI	U 00	MANIK_MB_Ana0...	156.29	0.00	Counts
!	4/6/2011 8:10:06	MANIK_MB_Asset	MANIK_MB_Ana029	RSHI	U 00	MANIK_MB_Ana0...	156.29	0.00	Counts
!	4/6/2011 8:10:06	MANIK_MB_Asset	MANIK_MB_Ana039	RSHI	U 00	MANIK_MB_Ana0...	156.29	0.00	Counts
!	4/6/2011 8:10:06	MANIK_MB_Asset	MANIK_MB_Ana049	RSHI	U 00	MANIK_MB_Ana0...	156.29	0.00	Counts
!	2/24/2011 15:21:04	MANIK_DB_ASSET	MANIK_DB_Ana050	RSHI	U 00	MANIK_DB_Ana0...	0.00	0.00	Atkos
⚠	2/24/2011 15:21:04	MANIK_DB_ASSET	MANIK_DB_Ana050	PVHI	H 00	MANIK_DB_Ana0...	0.00	0.00	Atkos
!	2/24/2011 15:20:13	MANIK_SCADA_Asset	MANIK_DB_ANA139	RSLO	U 00	ALGO#22 PIECE...	0.00	0.00	

Figure 22: Typical alarm table

Event tables

An *event table* lists events in a manner similar to the Event Summary.

However, unlike alarm, alert, and message tables, 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 *Server and Client Configuration Guide*.

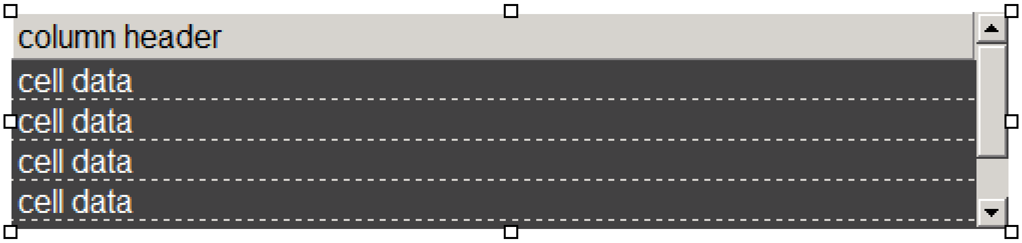
The properties you can specify when creating an event table include:

- The types of events that are shown (For example, you may only want to show events in the last 24 hours for a particular location.)
- The details that are shown (date/time, description, and so on), and the order in which they appear across the table
- Whether there are horizontal and vertical scroll bars

Event tables only display those events generated on the current day. This cannot be changed.

Notes

- When you create a 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 specify in the tables properties.



column header
cell data
cell data
cell data
cell data

Figure 23: Typical table as it appears in HMIWeb Display Builder

Related topics

"Alarm Table display object" on page 175
"Alert Table display object" on page 179
"Event Table display object" on page 188
"Message Table display object" on page 193
"Creating an alarm, alert, event, or message table" on page 177
"Sorting a table" on page 177
"Filtering a table" on page 178
"Columns properties" on page 229

Creating an alarm, alert, event, or message table

To create an alarm, alert, event, or message table

1 Choose one of the following icons from the **ToolBox** tab or the **Toolbox** toolbar.

To create…	…click
an Alarm table	▦⚠
an Alert table	▦!
an Event table	▦
a Message table	▦✉

2 Draw the table on the display.

3 Use the **Properties** tab or the **Properties** dialog box to show or hide columns in the table and to modify the properties and layout of the table.

To show or hide table columns

1 Display the **Data** tab on the table's **Properties** dialog window.

2 To show the column, select the checkbox next to the column name. To hide the column from view, clear the checkbox next to the column name.

3 Change the order in the list of the selected column by clicking **Move up** and **Move down**.

4 Enter the width of the column in pixels.

Related topics

"Sorting a table" on page 177

"Filtering a table" on page 178

"About alarm, alert, event, and message tables" on page 175

"Columns properties" on page 229

Sorting a table

You can apply up to three levels of sorting criteria on an Alarm Table, an Alert Table, a Message Table, and an Event Table.

To sort a table

1 On the `Sorting` tab of the table's properties, select a column from the **Sort by this column** list.

This is the primary sort column.

a Click **Ascending** or **Descending** to specify the sort order.

For example, you can sort columns by **Changed Time**, in ascending order. This means that items are listed in order of ascending time, that is, the oldest item is listed at the top of the summary.

2 To include a second sort column, select a column from the **Then by** list.

a Click **Ascending** or **Descending** to specify the sort order.

3 To include a third sort column, select a column from the **Then by** list.

a Click **Ascending** or **Descending** to specify the sort order.

Results

Data in the table sorts by the primary sort column, and then if specified, by the secondary and tertiary sort columns.

Attention

Preview the display (**File > Preview**) to ensure that table column widths are adequate for the expected content length.

In some cases, the column may truncate data if the column is too narrow. If necessary, increase the width of the column (usually by only a few pixels) on the table's **Data** tab.

Related topics

"Creating an alarm, alert, event, or message table" on page 177

"Filtering a table" on page 178

"About alarm, alert, event, and message tables" on page 175

"Columns properties" on page 229

Filtering a table

You can apply filters on an Alarm Table, an Alert Table, a Message Table, and an Event Table to include or exclude data. For example, you may want to show alarms on an Alarm Table that occurred only within the last hour.

To filter a table

1 On the `Filters` tab of the table's properties, select a column from the **Tick the columns you wish to filter by** list.

a Depending on the type of filter associated with the column, select the filter criteria.

Filter type	Description
Priority	Select one or more priorities you want to show.
State Indication	Select one or more states you want to show.
String	1. From the **Show entries where column value** list, select **equals** or **does not equal**. 2. Enter text into the box that you want to match or exclude. Use the question mark wildcard character **?** to represent a single character. Use the asterisk wildcard character ***** to represent multiple characters.
Time	Select an appropriate period from the **Time filter type** list. Note: Time filters are not available for Event Tables.

2 Repeat step 1 to add as many additional column filters as you need.

Related topics

"Creating an alarm, alert, event, or message table" on page 177

"Sorting a table" on page 177

"About alarm, alert, event, and message tables" on page 175

"Columns properties" on page 229

Alert Table display object

Description

Creates a specialized table that lists alerts.

Properties

- “Colors properties” on page 228
- “Columns properties” on page 229
- “Details (Alert Table and Message Table) properties” on page 240
- “Font properties” on page 248
- “General (Display object) properties” on page 249
- “Lines properties” on page 254

Related topics

“Alarm Table display object” on page 175
“Event Table display object” on page 188
“Message Table display object” on page 193
“About alarm, alert, event, and message tables” on page 175

Alphanumeric display object

Description

Displays point parameter or database values.

Properties

- “Animation properties” on page 218
- “Behaviors properties” on page 224
- “Colors properties” on page 228
- “Data properties” on page 232
- “Font properties” on page 248
- “General (Display object) properties” on page 249
- “Lines properties” on page 254
- “Presentation (Alphanumeric) properties” on page 260
- “Shortcut properties” on page 268

Arc display object

Description

Creates an arc (a quarter of an oval or circle).

Properties

- “Animation properties” on page 218
- “Behaviors properties” on page 224
- “Colors properties” on page 228
- “Data properties” on page 232
- “General (Display object) properties” on page 249
- “Lines properties” on page 254
- “Shortcut properties” on page 268

Basic Trend display object

Description

Creates a simple trend, which has fewer operator controls than a trend.

Properties

- "Colors properties" on page 228
- "Font properties" on page 248
- "General (Display object) properties" on page 249
- "Period properties" on page 256
- "Plot Details properties" on page 258
- "Shortcut properties" on page 268
- "View properties" on page 269

Related topics

"About trends and basic trends" on page 207

Bezier Curve display object

Description

Creates a smooth-curved line.

Properties

- “Animation properties” on page 218
- “Behaviors properties” on page 224
- “Colors properties” on page 228
- “Data properties” on page 232
- “General (Display object) properties” on page 249
- “Lines properties” on page 254
- “Shortcut properties” on page 268

Checkbox display object

Description

Creates a check box, which can be used to select or clear an option within a display.

Properties

- “Behaviors properties” on page 224
- “Colors properties” on page 228
- “Data properties” on page 232
- “Details (Checkbox) properties” on page 241
- “Font properties” on page 248
- “General (Display object) properties” on page 249
- “Shortcut properties” on page 268

Combobox display object

Description

Creates a combobox list, which users can select from a list of options within a display.

Context-sensitive enumeration

Where supported by the controller, combobox lists bound to a point parameter will display context-sensitive enumeration.

For example, if the PV values for a particular point parameter on a motor are ‘slow,’ ‘medium,’ and ‘fast,’ context-sensitive enumeration means that when the current value is ‘slow,’ then only ‘medium,’ and ‘fast’ will appear in the combobox. Likewise, when the current PV value is ‘medium,’ then only ‘slow,’ and ‘fast’ will appear, and so on.

Properties

- “Animation properties” on page 218
- “Behaviors properties” on page 224
- “Colors properties” on page 228
- “Data properties” on page 232
- “Font properties” on page 248
- “General (Display object) properties” on page 249
- “Lines properties” on page 254
- “Presentation (Combobox) properties” on page 264
- “Shortcut properties” on page 268

Display (display type)

Description

The display's basic properties, such as its title and description.

Properties

- “Appearance properties” on page 220
- “Callup Task properties” on page 227
- “Custom Properties properties” on page 231
- “General (Displays) properties” on page 250
- “Help properties” on page 252
- “Keys properties” on page 253
- “Page Details properties” on page 255
- “Associations - Alarm Group properties” on page 221
- “Associations - Multiwindow properties” on page 222
- “Periodic Task properties” on page 257
- “Shortcut properties” on page 268

Dynamic Shape

Description

A dynamic shape is a ‘custom object’ used in displays to present complex dynamic data.

Properties

- “Custom Properties properties” on page 231
- “General (Shapes) properties” on page 251

Event Table display object

Description

Creates a specialized table that lists events.

Properties

- “Colors properties” on page 228
- “Columns properties” on page 229
- “Details (Alarm Table and Event Table) properties” on page 239
- “Font properties” on page 248
- “General (Display object) properties” on page 249
- “Lines properties” on page 254

Related topics

“Alarm Table display object” on page 175
“Alert Table display object” on page 179
“Message Table display object” on page 193
“About alarm, alert, event, and message tables” on page 175

Faceplate (display type)

Description

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

Properties

- “Appearance properties” on page 220
- “Callup Task properties” on page 227
- “Custom Properties properties” on page 231
- “Faceplate properties” on page 246
- “General (Displays) properties” on page 250
- “Help properties” on page 252
- “Keys properties” on page 253
- “Page Details properties” on page 255
- “Associations - Multiwindow properties” on page 222
- “Periodic Task properties” on page 257
- “Shortcut properties” on page 268

Hyperlink display object

Description

Creates a hyperlink. When a user clicks on the hyperlink, the display calls up the specified URL, such as a web page or a display.

Properties

- “Behaviors properties” on page 224
- “Colors properties” on page 228
- “Details (Hyperlink) properties” on page 242
- “Font properties” on page 248
- “General (Display object) properties” on page 249
- “Shortcut properties” on page 268

Indicator display object

Description

Creates a gauge-like display object that shows a relative value in a graphical form.

Properties

- “Animation properties” on page 218
- “Behaviors properties” on page 224
- “Colors properties” on page 228
- “Data properties” on page 232
- “Details (Indicator) properties” on page 243
- “General (Display object) properties” on page 249
- “Lines properties” on page 254
- “Shortcut properties” on page 268

Line display object

Description

Creates a straight line.

Properties

- “Animation properties” on page 218
- “Behaviors properties” on page 224
- “Colors properties” on page 228
- “Data properties” on page 232
- “General (Display object) properties” on page 249
- “Lines properties” on page 254
- “Shortcut properties” on page 268

Message Table display object

Description

Creates a specialized table that lists messages.

Properties

- “Colors properties” on page 228
- “Columns properties” on page 229
- “Details (Alert Table and Message Table) properties” on page 240
- “Font properties” on page 248
- “General (Display object) properties” on page 249
- “Lines properties” on page 254

Related topics

“Alarm Table display object” on page 175
“Alert Table display object” on page 179
“Event Table display object” on page 188
“About alarm, alert, event, and message tables” on page 175

Oval display object

Description

Creates an oval or circle.

Properties

- “Animation properties” on page 218
- “Behaviors properties” on page 224
- “Colors properties” on page 228
- “Data properties” on page 232
- “General (Display object) properties” on page 249
- “Lines properties” on page 254
- “Shortcut properties” on page 268

Picture display object

Description

Inserts a picture (graphic).

Properties

- “Behaviors properties” on page 224
- “Details (Picture) properties” on page 244
- “General (Display object) properties” on page 249
- “Shortcut properties” on page 268

Polygon display object

Description

Creates a polygon.

To create a polygon, click to create each node in the polygon, except for the last node; you must double-click to create the last node, which closes the polygon.

Properties

- “Animation properties” on page 218
- “Behaviors properties” on page 224
- “Colors properties” on page 228
- “Data properties” on page 232
- “General (Display object) properties” on page 249
- “Lines properties” on page 254
- “Shortcut properties” on page 268

Polyline display object

Description

Creates a multi-segment line. Unlike a polygon, a polyline does not have a fill color, and can have an open side.

To create a polyline, click to create each node in the polyline, except for the last node; you must double-click to create the last node, which completes the polyline.

Properties

- “Animation properties” on page 218
- “Behaviors properties” on page 224
- “Colors properties” on page 228
- “Data properties” on page 232
- “General (Display object) properties” on page 249
- “Lines properties” on page 254
- “Shortcut properties” on page 268

Popup (display type)

Description

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

Properties

- “Appearance properties” on page 220
- “Callup Task properties” on page 227
- “Custom Properties properties” on page 231
- “General (Displays) properties” on page 250
- “Help properties” on page 252
- “Keys properties” on page 253
- “Page Details properties” on page 255
- “Associations - Multiwindow properties” on page 222
- “Periodic Task properties” on page 257
- “Shortcut properties” on page 268

Pushbutton display object

Description

Creates a button, which users can click to perform a specified command.

Properties

- “Behaviors properties” on page 224
- “Button Details properties” on page 225
- “Colors properties” on page 228
- “Font properties” on page 248
- “General (Display object) properties” on page 249
- “Lines properties” on page 254
- “Shortcut properties” on page 268

Rectangle display object

Description

Creates a rectangle or square.

Properties

- “Animation properties” on page 218
- “Behaviors properties” on page 224
- “Colors properties” on page 228
- “Data properties” on page 232
- “General (Display object) properties” on page 249
- “Lines properties” on page 254
- “Shortcut properties” on page 268

Rounded Rectangle display object

Description

Creates a rectangle or square with rounded corners.

Properties

- “Animation properties” on page 218
- “Behaviors properties” on page 224
- “Colors properties” on page 228
- “Data properties” on page 232
- “General (Display object) properties” on page 249
- “Lines properties” on page 254
- “Shortcut properties” on page 268

Shape display object

Description

Inserts a shape sequence or dynamic shape.

Properties

- “Behaviors properties” on page 224
- “General (Display object) properties” on page 249
- “Shape Details properties” on page 267
- “Shortcut properties” on page 268

Shape Sequence

Description

A shape sequence is a 'custom object' that is used in displays as either a status indicator or an animation.

Properties

- “Custom Properties properties” on page 231
- “General (Shapes) properties” on page 251

Table display object

Description

Creates a table within a display.

Properties

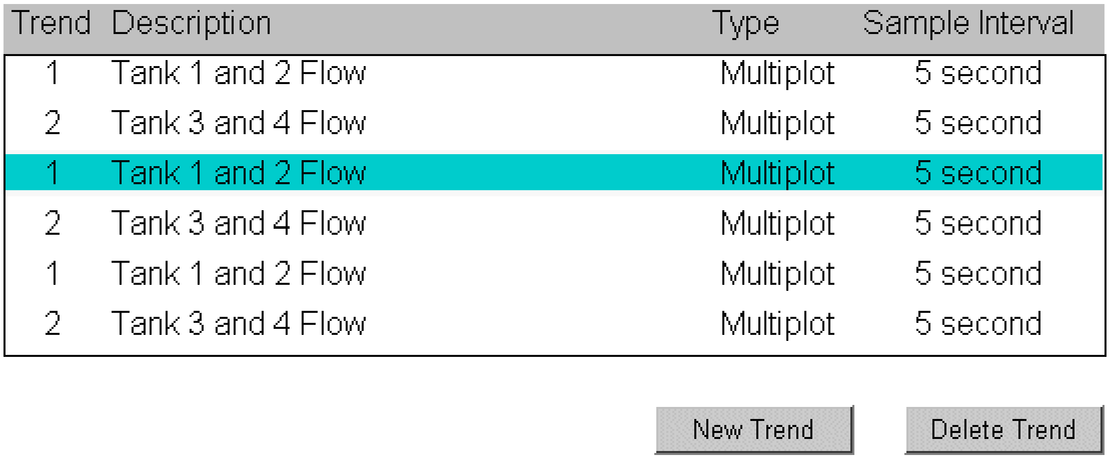
- "Colors properties" on page 228
- "Columns properties" on page 229
- "Details (Table) properties" on page 245
- "Font properties" on page 248
- "General (Display object) properties" on page 249
- "Lines properties" on page 254

About tables

A *table* lists data in a tabular format. The data can include alphanumeric, check box, push button, and combo box objects.

The properties you can specify when creating a table include:

- Whether the header row is visible
- Whether the table can be resized
- Whether there are horizontal and vertical scroll bars



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

Figure 24: Example table

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, clear the **Visible** checkbox.

7 Enter the width of the column in pixels.

8 To make the column a fixed size, clear 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.

Textbox display object

Description

Creates block of text.

Properties

- “Animation properties” on page 218
- “Behaviors properties” on page 224
- “Colors properties” on page 228
- “Data properties” on page 232
- “Font properties” on page 248
- “General (Display object) properties” on page 249
- “Lines properties” on page 254
- “Shortcut properties” on page 268

Trend display object

Description

Creates a trend, which displays process values over time in a graphical manner.

Properties

- “Colors properties” on page 228
- “Font properties” on page 248
- “General (Display object) properties” on page 249
- “Period properties” on page 256
- “Plot Details properties” on page 258
- “Shortcut properties” on page 268
- “View properties” on page 269

Related topics

“About trends and basic trends” on page 207

About trends and basic trends

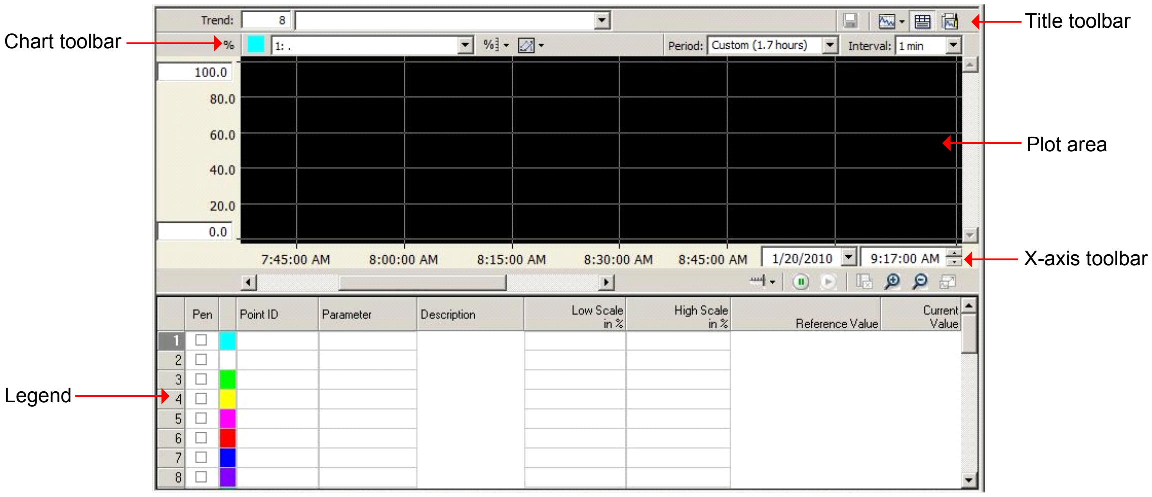
A trend object presents information in a graphical manner, and can be configured in numerous ways.

HMIWeb includes two types of trend object:

- **Trend**. A trend is a complex object with many components (as shown in the following figure) that allows you to give operators extensive control over how data is displayed in the trend.
- **Basic trend**. A basic trend is a much simpler object that only includes the plot area and the X-axis toolbar, which means that operators have much less control over the trend.

Notes

- Because of the complexity of the trend object, Honeywell strongly recommends that you only include one instance of it in a display. Use the basic trend object if you want to include several trends on a display.
- Some of the features described here can only be accessed using scripts. The Properties Window only allows you to configure a trend's basic properties.
- You can only create line and bar trends in HMIWeb Display Builder. If you want to create another type of trend, such as pie or tuning, you must use Display Builder.



Trend: 8

Period: Custom (1.7 hours) Interval: 1 min

1/20/2010 9:17:00 AM

	Pen		Point ID	Parameter	Description	Low Scale in %	High Scale in %	Reference Value	Current Value
1	□	🟦							
2	□	⬜							
3	□	🟩							
4	□	🟨							
5	□	🟪							
6	□	🟥							
7	□	🟦							
8	□	🟪							

Figure 25: The major components of the trend object

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 plot 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.

Related topics

"Basic Trend display object" on page 182

"Trend display object" on page 207

About copy and paste and drag and drop behaviors

When you copy and paste, or drag and drop, a display object onto a trend, the point parameter or parameters that are pasted or dropped onto the trend depend on the following:

- The value of the **Include in copy/paste** property of the display object.
- The copy and paste definitions within the shortcut menu for the point type.

'Include in copy/paste' property

The **Include in copy/paste** property for a display object specifies if a point parameter is pasted or dropped onto a trend during a copy and paste or drag and drop operation.

You set the **Include in copy/paste** property to **Exclude** when display objects are bound to point parameters which are not suitable for display in trends. For example, Alarm State display objects.

Copy and paste definitions in shortcut menus

Default shortcut menu definitions define the point parameters that can be copied and pasted or dragged and dropped.

Below is the basic element structure of a copy and paste definition within a shortcut menu.

```
<CopyPasteDefinitions>
  <CopyPasteDefaultParam>...</CopyPasteDefaultParam>
  <CopyPasteParam>  </CopyPasteParam>
</CopyPasteDefinitions>
```

The `<CopyPasteDefaultParam>` element contains a single point parameter, while the `<CopyPasteParam>` element can contain a list of comma separated point parameters.

It is only the point parameters included within this copy and paste definition that can be pasted or dropped onto a trend. If a user attempts to paste or drop a point parameter that is not included in this copy and paste definition, the default point parameter (as defined by the `<CopyPasteDefaultParam>` element) is pasted or dropped onto the trend.

If you copy or drag a group of display objects, or a Shape, and if the selected display object with the group or Shape is not bound to a point parameter, the remaining display objects within the group or Shape are scanned for bound point parameters. If there are other display objects within the group or Shape that are bound to point parameters, it is these point parameters that are pasted or dropped onto the trend; if these point parameters are included in the shortcut menu copy and paste definitions.

Related topics

"About shortcut menus" on page 132

"Shortcut properties" on page 268

"Data properties" on page 232

Plot area

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

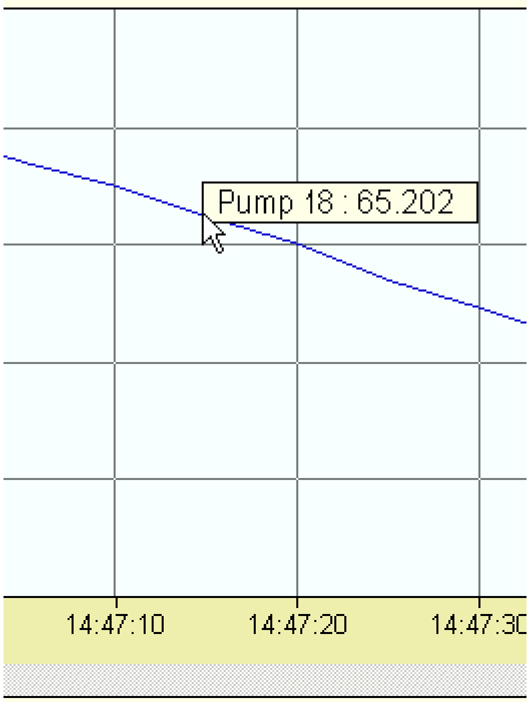
You can also configure the view's appearance, such as the background and grid colors.

Sampling and sampling intervals

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

Plot ToolTips

When a user moves the mouse pointer over a plot, a default mouse pointer ToolTip shows the value of the plot at that position. With plot data ToolTip formatting, you can customize exactly what is displayed in the mouse pointer ToolTip over a plot, such as including a suitable prefix to the value, as shown in the figure below.



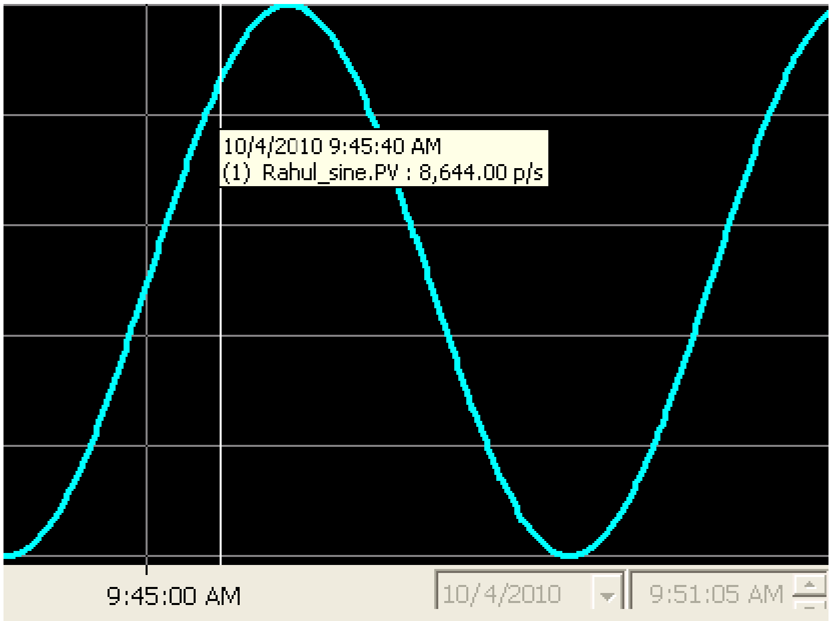
You can customize the plot data ToolTip using script to set the Format and DateFormat properties of the "PlotDataTip object" on page 383 in the HMIWeb object model. For details of what can be displayed in a plot data ToolTip, see "PlotDataTip object" on page 383 in the reference section.

Reference line and reference values

When a user clicks in the plot area, a vertical *reference line* appears. Only one reference line can be placed per trend. The exact placement of the reference line depends on whether the snap-to-value setting for the “ReferenceCursor object” on page 387 has been set or not. If the snap-to-value property is set, the reference line is placed at the nearest actual value in the trend. If snap-to-value is not set, the reference line passes through the point marked by the click. Also, when the user moves the pointer over the reference line, a reference line data ToolTip appears, which by default shows the date/time (X axis value) of the reference line followed by a list of visible (enabled) plots and their values at that time.

With scripting, you use the “OnReferenceCursorSet event” on page 530 to signal when to obtain the *reference values*. A reference value is the value (and date/time) of a plot where it intersects the reference line.

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



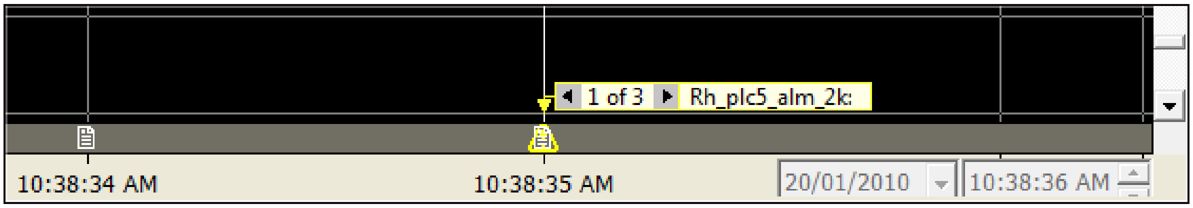
10/4/2010 9:45:40 AM
(1) Rahul_sine.PV : 8,644.00 p/s

Event bar and data tips

Attention

Event bar and data tips are not available for basic trends.

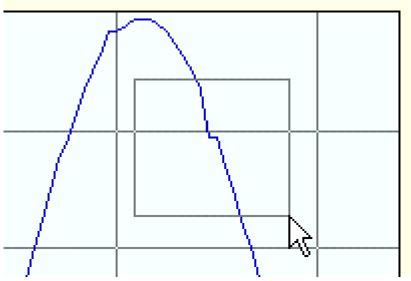
If the Event Summary is visible with the trend, an event bar is added to the bottom of the plot area. The event bar contains icons to indicate when an event occurred. A data tip appears when a user clicks on an event symbol. To remove the data tip a user clicks on the event symbol again. If a different 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.



1 of 3 Rh_plc5_alm_2k:

Zoom function

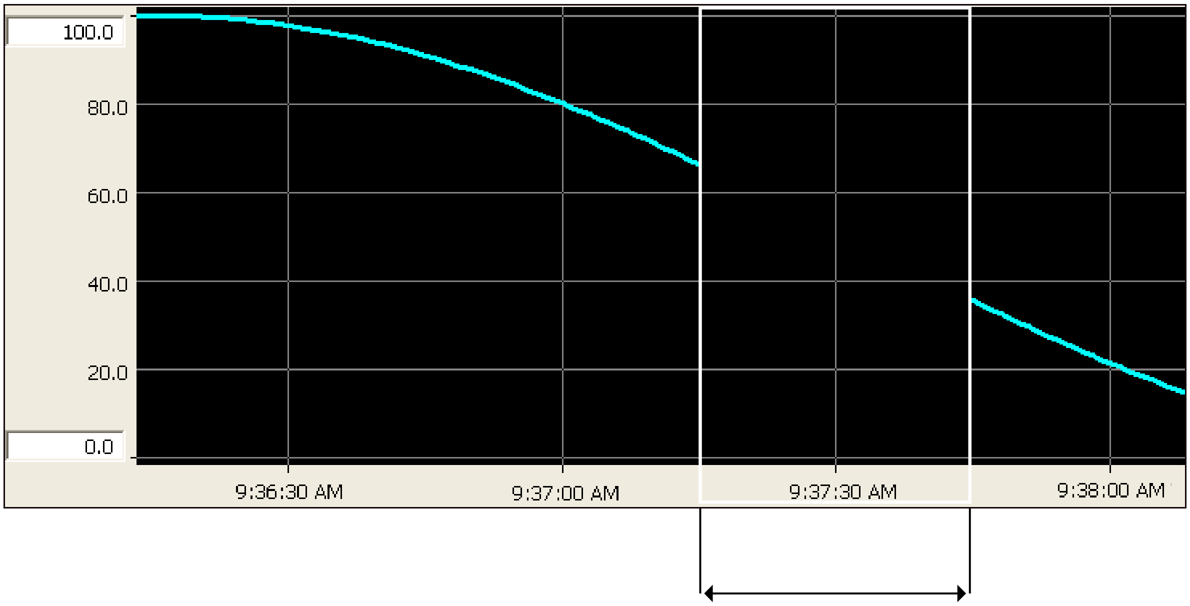
A user can zoom into a particular part of the trend by dragging diagonally across the area of interest, as shown in the following figure. When the user releases the mouse button, the trend zooms into the rectangle marked by the pointer.



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

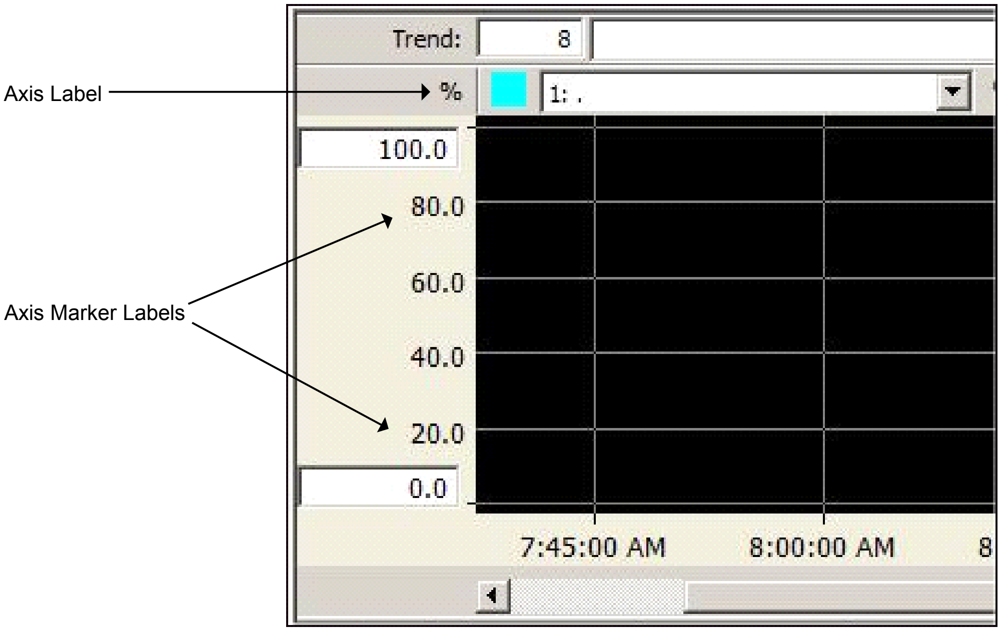
Axis

By default, the Axis View for the Y axis includes the following:

- The axis marker labels.
- The axis label.

  This appears on the Y axis if the toolbar is not visible or if using the basic trend. The axis label displays the engineering units of the point or a % symbol if in percent mode.

By default, the Axis View for the X axis includes the axis marker labels, however, the axis label (for example, Time) is not displayed.



Labels with multiple plots

If you are using individual scales for each plot on a trend:

- Each plot can have its own axis and marker labels.
- The scale of the current plot is displayed. The default plot that is displayed on custom trends is the first plot, however, this can be changed.

If you are using a single scale for all plots on a trend, only one scale is displayed.

Toolbars

A trend (as opposed to a basic trend) has several toolbars that you can make visible, which therefore allows you to determine the level of control operators have over what appears in the trend. A basic trend only includes the X-axis, which you can hide if you do not want to give operators any control over it.

Operators can, for example, use the toolbars to:

- Change the period and sample interval.

  You use the axis to 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” on page 489 to specify what items are available on the toolbar.

Legend

Attention

Legends are not available for basic trends.

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.

Wedge display object

Description

Creates a segment (quarter) of an oval or a circle. Unlike an Arc display object, a Wedge display object has a fill color.

Properties

- “Animation properties” on page 218
- “Behaviors properties” on page 224
- “Colors properties” on page 228
- “Data properties” on page 232
- “General (Display object) properties” on page 249
- “Lines properties” on page 254
- “Shortcut properties” on page 268

Display and display object properties

This section describes display object properties.

To view the **Properties** pane, choose **View** > **Properties Grid**.

The following topics describe the properties for display objects.

Related topics

“ActiveX Properties properties” on page 217
“Animation properties” on page 218
“Appearance properties” on page 220
“Associations - Alarm Group properties” on page 221
“Associations - Multiwindow properties” on page 222
“Behaviors properties” on page 224
“Button Details properties” on page 225
“Callup Task properties” on page 227
“Colors properties” on page 228
“Columns properties” on page 229
“Custom Properties properties” on page 231
“Data properties” on page 232
“Details (Alarm State) properties” on page 237
“Details (Alarm Table and Event Table) properties” on page 239
“Details (Alert Table and Message Table) properties” on page 240
“Details (Checkbox) properties” on page 241
“Details (Hyperlink) properties” on page 242
“Details (Indicator) properties” on page 243
“Details (Picture) properties” on page 244
“Details (Table) properties” on page 245
“Faceplate properties” on page 246
“Font properties” on page 248
“General (Display object) properties” on page 249
“General (Displays) properties” on page 250
“General (Shapes) properties” on page 251
“Help properties” on page 252
“Keys properties” on page 253
“Lines properties” on page 254
“Page Details properties” on page 255
“Period properties” on page 256
“Periodic Task properties” on page 257
“Plot Details properties” on page 258
“Popup properties” on page 259

"Presentation (Alphanumeric) properties" on page 260
"Presentation (Combobox) properties" on page 264
"Script Data properties" on page 265
"Shape Details properties" on page 267
"Shortcut properties" on page 268
"View properties" on page 269

ActiveX Properties properties

The properties available for an ActiveX control are dependent on the ActiveX control inserted into the display.

For more information about the properties available for an ActiveX control, see the documentation from the developer of the ActiveX control.

Animation properties

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

Property	Description
Color Breakpoint Properties	
Breakpoint Type	The type of color breakpoint to use for this 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. • **Discrete** = Each color represents a particular discrete value. Note that only eight discrete animation states are available, so this breakpoint type should not be used with parameters larger than eight bits (such as ALMSTS).
Range *n*	
Range *x* TO *y*	This property is only available if **Continuous** is the selected value for the **Breakpoint Type** property. For example, you could make an object turn red when the value is between 90 and 100 percent of its range.
Color	This property is only available if **Continuous** is the selected value for the **Breakpoint Type** property. The display color for when the value is within the specified range.
State *n*	
Inverse	This property is only available if **Discrete** is the selected value for the **Breakpoint Type** property. Select **True** 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.
Color	This property is only available if **Discrete** is the selected value for the **Breakpoint Type** property. The display color for when the value is in the specified state.
Indicate NaN with color	Possible Values are **True** or **False**. If **True**, 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 Bad value with color	Possible Values are **True** or **False**. If **True**, Station overrides the current color with the system-defined color for 'off-scan'. The default color is gray.
Indicate Alarm state with color	Possible Values are **True** or **False**. If **True**, Station overrides the object's default color with the 'aggregated alarm state' color if the associated point has one or more alarms. For example, if the point had these two alarms: • Urgent, Acknowledged, Active • High, Unacknowledged, Returned-to-Normal The object will flash red because: • Urgent is the highest priority of all the alarms (hence color is red) • One of the alarms is unacknowledged (hence flashing) • One of the alarms is active (show color) In practice, there will be no indication of an alarm if the value falls to zero. If the alarm colors have been customized and the Use these colors for points on displays setting has been used, then the colors described here are overridden with the customized alarm colors.

Property	Description
Indicate Off-scan state with color	Possible Values are **True** or **False**. If **True**, 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.

Appearance properties

The Appearance properties define the style of the display.

Property	Description
Style	A list of style class definitions contained within the stylesheet specified in the **Stylesheet** property. Only class definitions that are appropriate for displays appear in this list.
Image	Specifies a background image for the display. If you want the image to fill the background of the display, the image dimensions must be the same as the display size.
Stylesheet	The stylesheet which contains the style definitions to use for this display.
Size	
Width	The width of the display, in pixels.
Height	The height of the display, in pixels.
Color	The background color of the display.

Associations - Alarm Group properties

The Associations – Alarm Group properties control alarm group generation.

Property	Description
Associate with an alarm group	Possible values are **True** or **False**. If this property is set to **True**, this page will be associated with the alarm group as defined by the other properties (default or custom alarm group). If you are using tabbed displays, and you want an alarm icon to appear on the tab of custom displays when a point on that display is in an alarm state, you must associate the display with an alarm group. If this property is set to **False**, no alarm group is associated with the display.
Use default alarm group	This property is only available if **True** is the selected value for the **Associate with an alarm group** property. Possible values are **True** or **False**. If this property is set to **True**, the display is associated with an alarm group, and the name of that alarm group is based on the default alarm group name defined in the HMIWeb Display Builder options. If this property is set to **False**, the display is associated with the alarm group name defined by the **Custom alarm group** property.
Custom alarm group	This property is only available if **False** is the selected value for the **Use default alarm group** property. Type the alarm group name to associate with the display.

Associations - Multiwindow properties

The Associations – Multiwindow properties control multi-window page properties, such as Safeview.

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_Pnz_Display**. A Pan and Zoom display. • **HW_Pnz_Thumbnail**. The Pan and Zoom thumbnail • **HW_System_Trend**. A trend. • **HW_System_Detail**. A point detail display. • **HW_System_Group**. A group display. • **HW_System_Alarm_Summary**. The Alarm Summary. • **HW_System_Faceplate**. A faceplate. • **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 recalled when a user clicks ◀ . You can select a window category from the list or you can type a category into the list box. If you want to add a category to the list rather than type it each time, see the topic titled "Defining SafeView window categories" for information about how to add a new window category to the list of standard categories. Specialized features such as Digital Video Manager (DVM) can be restricted to show in a predefined window using SafeView window category 'DVM.'
Single use application set	Enable this option if you want to ensure that certain related displays are not shown simultaneously, and then use the following box to specify those displays.
Application set name	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. Only applicable if the display is to be used by a multi-window Station.

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 Associations – Multiwindow properties grid are defined in the file titled *WindowCategories.xml*. (This is an XML file, located in *<install folder>\Honeywell\Experion PKS\Client\HMIWeb Display Builder*, where *<install folder>* is the location where Experion is installed.)

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>
```

Prerequisites

1. You must have Windows administrator permissions to perform this task.

To add a new window category to the SafeView window category list

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 of items in the XML file controls the order in which items appear in the list.
3. Edit the pasted category entry as follows:
   The following example shows how to add a window category named *DVM*.

   ```
   <category name="DVM"/>
   ```
4. Save and close the file.

Next steps

If *HMI Web Display Builder* is currently open, close and relaunch the application to see the updated list.

Behaviors properties

The Behaviors properties specify which behaviors are attached to the display object. Some behaviors are not applicable to all display objects.

Behavior	Description
Faceplate	Possible Values are **True** or **False**. If **True**, 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.)
Hover	Possible Values are **True** or **False**. If **True**, only applicable to Alphanumeric and Combobox display objects. 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. **Attention** If you select this behavior, do not use the object's **ToolTip** property (General properties).
Popup	Possible Values are **True** or **False**. If **True**, a popup appears when a user clicks the object. Adds the Popup properties, in which you specify the popup's properties.
Script Data	See the Script Data properties to specify the point parameters that can be accessed by scripts.

Related topics

"Popup properties" on page 259

"Script Data properties" on page 265

"General (Display object) properties" on page 249

Button Details properties

Property	Description
Label	The button's label.
Image	The image (graphic) that forms the button's background.
Action	The action performed when a user clicks the button. The options are: • **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 more information, see the *Server and Client Configuration Guide*. • **Request Report** = Request the specified report. • **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. • **Current Display to** = Applicable only to multiple static Station. Send the current display to the specified Station within the console. You specify the Station in the Location box. • **Next Display to** = Applicable only to multiple static Station. Send the next display to the specified Station within the console. You specify the Station in the Location box. For more information about the Server Display Program (LRN 21) actions and parameters, see the *Server and Client Configuration Guide*. Select **None** if you want to write a script for the button's OnClick event.
Callup Page	This property is only available if **Callup Page** is the selected value for the **Breakpoint Type** property.
Report Number	This property is only available if **Request** is the selected value for the **Breakpoint Type** property.
Security Level	This property is only available if **Request Task** is the selected value for the **Breakpoint Type** property. The security level required to perform the task.
Task LRN	This property is only available if **Request Task** is the selected value for the **Breakpoint Type** property. The LRN number (task identifier).
Add station number offset to LRN	This property is only available if **Request Task** is the selected value for the **Breakpoint Type** property. **Attention** This property is deprecated and is provided only for legacy systems that have made use of it. If you set the **Add station number offset to LRN** property to **True**, 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.
Parameter 1	This property is only available if **Request Task** is the selected value for the **Breakpoint Type** property. The first parameter for the specified LRN, if required.

Property	Description
Parameter 2	This property is only available if **Request Task** is the selected value for the **Breakpoint Type** property. The second parameter for the specified LRN, if required.
Parameter 3	This property is only available if **Request Task** is the selected value for the **Breakpoint Type** property. The third parameter for the specified LRN, if required.
Parameter 4	This property is only available if **Request Task** is the selected value for the **Breakpoint Type** property. The fourth parameter for the specified LRN, if required.
Location	This property is only available if **Current Display to** or **Next Display to** is the selected value for the **Breakpoint Type** property.

Callup Task properties

The Callup Task properties specify 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.

Property	Description
Request task on page callup	Select this to specify a call up task.
Security Level	The security level required to perform the task.
Task LRN	The LRN number (task identifier).
Add station number offset to LRN	! **Attention** This property is deprecated and is provided only for legacy systems that have made use of it. If you set the **Add station number offset to LRN** property to **True**, 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.
Parameter 1	The first parameter for the specified LRN, if required.
Parameter 2	The second parameter for the specified LRN, if required.
Parameter 3	The third parameter for the specified LRN, if required.
Parameter 4	The fourth parameter for the specified LRN, if required.

Notes

- Do not callup a periodic task in the Callup Task properties of a display. If you want to perform the task at regular intervals while the display is visible, use the Periodic Task properties.
- Do not callup the Server Display Program (LRN 21) task in the Callup Task properties of a display. It does not make sense to call up another page on display call up.
- For more information about displaying the LRN assigned to an application program, see the *Application Development Guide*.

Related topics

"Periodic Task properties" on page 257

Colors properties

The Colors properties control the display object's colors.

The number of color properties depends on the display object type. For example, a Rectangle display object has line and fill colors, whereas an Alphanumeric display object also has a text color.

Property	Description
Fill Color	The object's fill color.
Fill Color Blink	Possible values are **True** or **False**. If **True**, the object's fill color blinks on and off.
Line Color	The object's line color.
Line Color Blink	Possible values are **True** or **False**. If **True**, the object's line color blinks on and off.
Text Color	The text color. Only applicable to objects with text, such as Alphanumeric and Pushbutton display objects.
Text Color Blink	Possible values are **True** or **False**. If **True**, the object's text color blinks on and off.
Level Fill Color	The color of the bar or pointer used to indicate the value. Only applicable to indicator objects.
Level Fill Blink	Possible values are **True** or **False**. If **True**, the indicator object's bar or pointer's color blinks on and off.

Trend and Basic Trend properties

Property	Description
X axis	
Background	The background color of the X axis.
Marker labels	The color of the X axis markers.
Axis labels	The color of the X axis labels.
Y axis	
Background	The background color of the Y axis.
Marker labels	The color of the Y axis markers.
Axis labels	The color of the Y axis labels.
Legend labels	
Background	The background color of the legend.
Text	The text color of the legend.

Columns properties

The **Columns** properties contains the following tabs.

Tab	Description
Data	Shows or hides columns for the selected table, changes the order of the columns, and sets the width for each column.
Sorting	Sorts by one or more columns. See the topic titled *Sorting a table* in the *HMIWeb Display Building Guide*.
Filters	Filters by one or more columns. See the topic titled *Filtering a table* in the *HMIWeb Display Building Guide*.

Data tab

The **Data** tab controls which columns appear in the table, and specifies their placement and width in the table.

Item	Description
Columns	Shows or hides columns. See the matrix below for which columns are available for each table type.
Move up	Moves the selected column one position higher in the list.
Move down	Moves the selected column one position lower in the list.
Width	The width (in pixels) of the selected column.

Columns matrix

This matrix shows which columns are available for display, based on the type of table selected.

Note: Items are listed here alphabetically. However, the order of items listed may vary on each table’s property sheet.

Column	Alarm Table	Alert Table	Event Table	Message Table
Access Reason	Y		Y	
Accessibility	Y	Y		
Action	Y	Y	Y	Y
Alarm Limit	Y		Y	
Alarm State Indication	Y			
Alarm Time	Y			
Alert Limit		Y		
Alert State Indication		Y		
Alert Time		Y		
Author	Y	Y		
Block	Y		Y	Y
Card Holder First Name	Y		Y	
Card Holder Last Name	Y		Y	
Card Number	Y		Y	
Category	Y			Y
Changed Time	Y	Y		Y

Column	Alarm Table	Alert Table	Event Table	Message Table
Classification		Y		
Comments Added		Y		
Condition	Y	Y	Y	Y
Count		Y		
Description	Y	Y	Y	
DSA Connection Name		Y		Y
DSA Server Name		Y		Y
Expiry Time		Y		
Field Time	Y	Y		Y
Item	Y	Y	Y	Y
Live Value		Y		
Location	Y	Y	Y	Y
Location Item	Y	Y	Y	Y
Location Tag	Y	Y	Y	Y
Message				Y
Message Limit				Y
Message State Indication				Y
Message Time				Y
OPC Severity	Y	Y	Y	Y
Operator	Y	Y	Y	Y
Previous Value	Y	Y	Y	Y
Priority	Y		Y	
Quality	Y	Y	Y	Y
Reason	Y	Y	Y	Y
Signature Meaning	Y		Y	Y
Source	Y	Y	Y	Y
Station	Y	Y	Y	Y
Sub-Condition	Y	Y	Y	Y
TPS Aux Unit	Y		Y	Y
TPS TS Reliability	Y		Y	Y
Trip Value		Y		
Units	Y	Y	Y	Y
Value	Y		Y	Y

Related topics

"Creating an alarm, alert, event, or message table" on page 177
"Sorting a table" on page 177
"Filtering a table" on page 178
"About alarm, alert, event, and message tables" on page 175

Custom Properties properties

The Custom Properties properties define your own properties for a dynamic shape or shape sequence.

To add a custom property, click **Add**.

Property	Description
Name	The custom property's name, which must not contain '&' or '?' characters.
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 “Example: Using a shape custom property to store a user-defined value” on page 321.
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.
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 available for the life of the Station.
Description	A description of the custom property.

Data properties

The Data properties specify the object's link to the database (such objects are called *dynamic objects*).

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 “Points” section of the *Server and Client Configuration Guide*.)

Property	Description
Database link type	The type of link the object has to the database: • **None** = The object is not linked to anything. (In the case of a combo box, for example, you would select this if you wanted to populate the combo box using a script.) • **Point/Parameter** = The object is linked to a point parameter. • **Database file** = The object is linked to a file in the server database, whose contents are controlled by another application. • **Database index** = The object is linked to a file in the server database, whose contents are controlled by another application.
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 tab to browse points on this server or another server in the system.
Parameter	The point parameter to which the object is linked. 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` . For a list of the history parameters, see the “Summary of internal point parameters” topic in the “Points” section of the *Server and Client Configuration Guide*. ❗ **Attention** When you choose **Show point disabled icon** and/or **Show point journal only icon** (Details (Alarm State) properties), you must also change the **Parameter** value. See “Understanding alarm state icons for disabled and journal only points” on page 237 for more information.
Include in copy/paste	When copying and pasting or dragging and dropping this display object to a trend, this property specifies the copy and paste behavior of the point parameter that this display object is bound to. • **Inherit from point type** = Same as **Include**. • **Include** = The point parameter, which this display object is bound to, is pasted or dropped onto the trend, so long as the point parameter is included in the copy and paste definitions of shortcut menu for the point type. • **Exclude** = The point parameter, which this display object is bound to, is not pasted or dropped into a trend.
Parameter Offset	If you select a history parameter, such as *H24H*, **Parameter Offset** controls which history value is used. (**1**, the default, represents the latest history value, **2** represents the second-to-last value, and so on.)
Parameter Indexing	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.

Property	Description
Offset	Select the type of index offset: • **Current file offset** = Adds Station’s current file offset to the file number. The offset can be set by scroll bars, by the server, and by the PAGE UP and PAGE DOWN keys. • **Current record offset** = Adds Station’s current record offset to the record number. The offset can be set by scroll bars, by the server, and by the PAGE UP and PAGE DOWN keys • **Current field offset** = Adds Station’s current field offset to the word number. The offset can be set by scroll bars, by the server, and by the PAGE UP and PAGE DOWN keys.
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**. **Attention** • The actual update rate is also determined by the Station and display update rates. For more information, see the "Understanding update rates" topic in the "Customizing Stations" section of the *Server and Client Configuration Guide*. • Dynamic scanning configuration for accumulator, analog, and status point parameters affects the update rate. See the "About dynamic scanning" topic in the "Points" section of the *Server and Client Configuration Guide*. • When using HMIWeb Display Builder to create custom displays with Fieldbus parameters, select an update rate of 30 seconds or faster. An update rate of 30 seconds is the optimal rate for FIM CPU consumption. • Update rates greater than 30 seconds for parameters on custom displays are not supported. This restriction is for custom displays only and not for parameters configured for trends or history collection, or accessed through OPC or FDM.
Allow fast update	Determines if the value will be updated at the fast update rate defined in the Station settings, when the FAST key is pressed on the IKB.
Data entry allowed for	Select a security level. 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**.

Related topics

“About copy and paste and drag and drop behaviors” on page 208

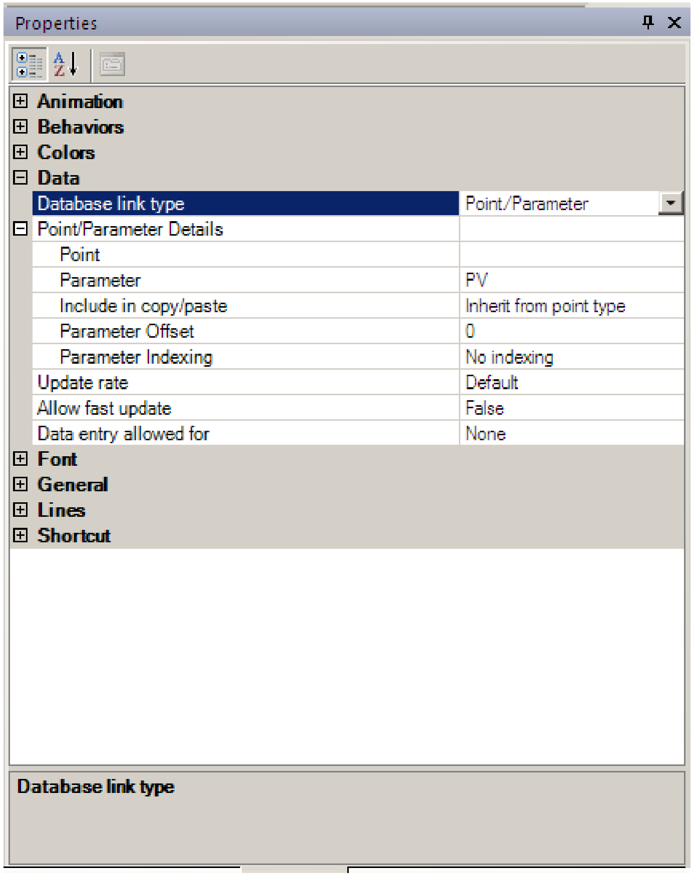
Defining a link to a point

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.

Tip

To define a link to a single element in an array, use the [x] notation, for example point.param[x] or point.param[x][y]



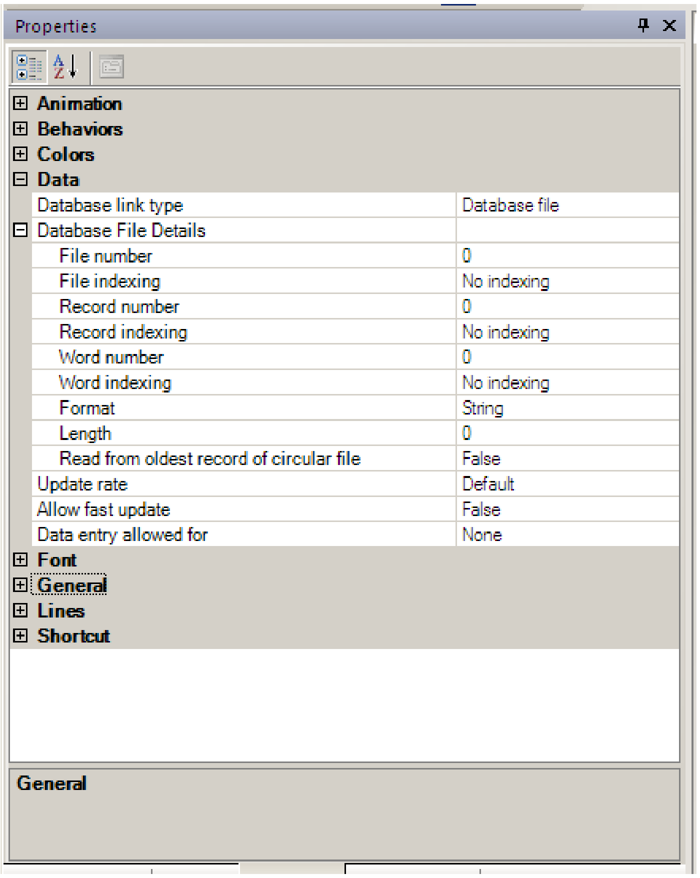
Database link type	Point/Parameter
⊟ Point/Parameter Details	
Point	
Parameter	PV
Include in copy/paste	Inherit from point type
Parameter Offset	0
Parameter Indexing	No indexing
Update rate	Default
Allow fast update	False
Data entry allowed for	None

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. An example of a full item name for a point called *`pump1`* might be *`/Assets/Digestion/Train1/Digestor/pump1`* (where *`Assets`* is the root directory and *`Digestion/Train1/Digestor`* is the asset hierarchy). Note that 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.
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`**. For a list of the history parameters, see the topic titled "Summary of internal parameters" in the "Points" section of the *Server and Client Configuration Guide*.
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.

Property	Description
Allow fast update	Determines if the value will be updated at the fast update rate defined in the Station settings, when the FAST key is pressed on the IKB.
Update rate	The frequency that the parameter value is updated: • `Default`. The value is updated at the Station update rate, which is 1 second. • `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`. **Attention** The actual update rate is also determined by the Station and display update rates. For more information, see “Understanding update rates” in the “Customizing Stations” section of the *Server and Client Configuration Guide*.
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*.



Properties

Database link type	Database file
⊟ Database File Details	
File number	0
File indexing	No indexing
Record number	0
Record indexing	No indexing
Word number	0
Word indexing	No indexing
Format	String
Length	0
Read from oldest record of circular file	False
Update rate	Default
Allow fast update	False
Data entry allowed for	None

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*.
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.
Allow fast update	Determines if the value will be updated at the fast update rate defined in the Station settings, when the FAST key is pressed on the IKB.
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*. ! **Attention** The actual update rate is also determined by the Station and display update rates. For more information, see “Understanding update rates” in the “Customizing Stations” section of the Server and Client Configuration Guide.
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*.

Details (Alarm State) properties

Property	Description
Show ToolTips	Specifies if tooltips are shown when you move the mouse over a toolbar button.
Show point disabled icon	Specifies if the disabled icon D appears when alarms on the point are disabled.
Show point journal only icon	Specifies if the Journal Only icon J appears when alarms on the point are journaled.
Make icon sizeable	Specifies if the icon fills the boundaries of the 'outline rectangle.' When selected, you can resize the icon to be larger than the default size of 22 × 22 pixels.

Attention

When you choose **Show point disabled icon** and/or **Show point journal only icon**, you must also change the **Parameter** value (Data properties). See "Understanding alarm state icons for disabled and journal only points" on page 237 for more information.

Understanding alarm state icons for disabled and journal only points

When creating an alarm state display object, you can choose to display an icon when alarms on the point are disabled or journal only. On the point's **Details** section of the **Properties** tab, the two options are:

- **Show point disabled icon**. The icon that displays on the point is D
- **Show point journal only icon**. The icon that displays on the point is J

You can choose both options. However, because the disabled state does not record any alarms, it takes precedence over the journal only state and the point disabled icon will appear.

The following table shows which icon will display in the following scenarios.

Option(s) selected	Point in alarm	Point alarms disabled	Point alarms journal only[3]	Point alarms disabled and journal only
Neither option	!, ⚠, or 🛡	No icon appears	No icon appears	No icon appears
Show point disabled icon only See "Updating the Parameter value to support Disabled icons and Journal Only icons" for more information.	!, ⚠, or 🛡	D	No icon appears	D
Show point journal only icon only See "Updating the Parameter value to support Disabled icons and Journal Only icons" for more information.	!, ⚠, or 🛡	No icon appears	J	J
Show point disabled icon and **Show point journal only icon** See "Updating the Parameter value to support Disabled icons and Journal Only icons" for more information.	!, ⚠, or 🛡	D	J	D

Updating the Parameter value to support Disabled icons and Journal Only icons

For all point types except TPS, Disabled and Journal Only states are stored in separate parameters. TPS points store the Journal Only state and the Disabled state in the same parameter. Therefore, for TPS points, it is only

[3] **Not applicable for TPS points.**

possible to show the Journal Only state if you have chosen to show the Disabled state; you cannot show the Journal Only state by itself.

When you choose **Show point disabled icon** and/or **Show point journal only icon**, you must also update the **Parameter** value (Data properties), otherwise the icon may not appear. The value you choose depends on the point type and the combination of Disabled/Journal Only states you want to show. Use the following table to determine the correct parameter value.

For this point type…	…to show point Disabled icon	…to show point Journal Only icon	…to show point Disabled and Journal Only icons
Process (except Foundation Fieldbus)	*almenbstate*	*JournalOnly*	*almenbstate*
Foundation Fieldbus	*alarm_sum.disabled*	*JournalOnly*	*alarm_sum.disabled*
SCADA	*AlarmDisabled*	*JournalOnly*	*AlarmDisabled*
TPS	*alenbst*	Not applicable	*alenbst*

Details (Alarm Table and Event Table) properties

Property	Description
Formatting Properties	
Resolution	Select the number of decimal places in milliseconds to display results to from the dropdown menu. The options are: **Seconds**, **10 Milliseconds**, **100 Milliseconds**, and **1000 Milliseconds**.
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.
Detail Properties	
Horizontal scrollbar	The horizontal scroll bar visibility setting. The values are: • **Hidden** = The scroll bar is hidden. • **Visible** = The scroll bar is visible. • **Auto** = The scroll bar only appears if the table is narrower than total width of the columns.
Vertical scrollbar	The vertical scroll bar visibility setting. The values are: • **Hidden** = The scroll bar is hidden. • **Visible** = The scroll bar is visible. • **Auto** = The scroll bar only appears if the table cannot simultaneously show all the alarms.
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.
Row height	The height (in pixels) of the rows.
Header visible	Possible values are **True** or **False**. If **True** is selected, the table includes column headings.
Header height	Specifies the height (in pixels) of the column headings.

Details (Alert Table and Message Table) properties

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

Details (Checkbox) properties

Property	Description
Use inverse logic	Reverses the standard check box logic, so that it corresponds to the wording of the **Label**. In standard logic, checked equals '1.' For example, labeling a check box 'Open' would require the opposite logic to labeling it 'Close.'
Label	The descriptive text that appears to the right of the check box.

Details (Hyperlink) properties

Property	Description
Page/URL	The destination address of a web page, FTP site or other Internet resource.
Action	Calls up the specified object: • **Callup URL** = Calls up the specified URL. • **Callup Page** = Calls up the specified display (which can be a DSP or HMIWeb display). Note that you do not have to specify the path because Station automatically searches its display folders. • **None** = Does nothing.

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
```

Details (Indicator) properties

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 solid pointer moves up and down within the object to indicate the relative value. • **Hollow Pointer** = A hollow pointer moves up and down within the object to indicate the relative value. • **Line** = A horizontal line moves up and down within the object to indicate the relative value. ! **Attention** If the data being sent to the Indicator is invalid, the Indicator will display a gray cross.
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.
Pointer Direction	Specifies the direction in which the indicator pointer faces. The options are **Left** or **Right**.
Range 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.
Range 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.

Details (Picture) properties

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

Details (Table) properties

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

Faceplate properties

Property	Description
Mode	Specifies how the mode element (or parameter) on the faceplate is identified. Choose: • **None** if the faceplate has no mode element. Note that auto-selection depends on the **Specify auto-selection behavior** setting. If this setting specifies that the SP or OP is to be selected based on the mode, then you must specify a mode element. • **Specify element via script** if the mode is determined by a script attached to the `OnRequestModeElement` event. You specify the Mode Element ID on the Page object's `modeElement` property. • From the list of elements currently on the faceplate to specify the mode element for this faceplate.
Set point	Specifies how the set point element (or parameter) on the faceplate is identified. Choose: • **None** if the faceplate has no set point element. Note that if there is no SP, then the SP cannot be auto-selected when the mode is AUTO. Furthermore, the **Select SP** command will not work. • **Specify element via script** if the set point is determined by a script attached to the `OnRequestSetPointElement` event. You specify the Set Point Element ID on the Page object's `setpointElement` property. • From the list of elements currently on the faceplate to specify the set point element for this faceplate.
Output	Specifies how the output element (parameter) is identified. Choose: • **None** if the faceplate has no output element. • **Specify element via script** if the output is determined by a script attached to the OnRequestOutPutElement event. You specify the Output Element ID in the Page object's **outputElement** property. • From the list of elements currently on the faceplate to represent the output for this faceplate.
Auto selection behavior	The default setting **Select SP or OP based on Mode** ensures that auto-selection on this faceplate is based on mode. When the faceplate's mode is: • AUTO, the set point (SP) is auto-selected. • MAN, the output (OP) is auto-selected. For consistency with other faceplates, it is recommended that you do not change this default setting unless this faceplate behaves very differently to others (for example, if there is no mode element on the faceplate). Choose **None** if you do not want auto-selection enabled on this faceplate. This setting overrides the auto-selection setting enabled on the server. Note that if you do not want auto-selection to operate in your system at all, then you should disable auto-selection on the Server Wide Settings display. Choose **Select specified element** if you want to nominate an element for auto-selection when a faceplate is called up.
Element	If you choose one of the available elements listed in the **Auto selection behavior** property, that element is selected when the faceplate is called up. This selection overrides any mode setting for this faceplate. Alternatively, you can choose **Specify element via script** to have an `OnRequestAppropriateElement` event fired when the faceplate is called up.
Command Element	Indicates which element describes the faceplate.

Notes

The Faceplate properties specify the point's control parameters for the faceplate. The parameters are specific to the type of point the faceplate represents.

- Auto-selection on faceplates is enabled via the Server Wide Settings display (see the "Server wide settings" topic in "Customizing Stations" section of the *Server and Client Configuration Guide*.
- For information on configuring faceplates, see the "Configuring faceplates" section of the *Server and Client Configuration Guide*. For an introduction on how the parameters on the faceplate are used, see the "Faceplate auto-selection" topic in the "Configuring faceplates" section of the *Server and Client Configuration Guide*.

Font properties

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

Property	Description
Bold	Possible values are **True** or **False**. If **True**, the text will appear in **bold**.
Italic	Possible values are **True** or **False**. If **True**, the text will appear *italicised*.
Strikeout	Possible values are **True** or **False**. If **True**, draws a line through the text.
Underline	Possible values are **True** or **False**. If **True**, draws a line below the text.
Alignment	The horizontal alignment of text within the object.
Name	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.)

General (Display object) properties

The General properties control 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. HMIWeb Display Builder gives the object a default name when you create the object, such as 'alpha003' for the third alphanumeric added to the display. If you change the name, you must ensure that it is unique— both with respect to other objects in the display and names such as custom properties and point parameters. Naming is particularly important if you intend to write scripts for the display.
ToolTip	This property does not apply for Trend display objects. The text that appears when a user hovers the mouse pointer over the object. **Attention** Leave this blank if you select the Hover property in the Behaviors properties.
Position	
Left Top	The object's position from the top left of the display, in pixels.
Size	
Width Height	The object's size, in pixels.
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.)
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** in the display's Appearance properties.)
Tab stop	Possible **True** or **False**. If **True** is selected, users can select the object by pressing TAB.
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.

Related topics

“Behaviors properties” on page 224

General (Displays) properties

The General properties control the display's basic properties such as its title and description.

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.

General (Shapes) properties

The General properties control a shape's basic properties, such as title, description, and stylesheet.

Property	Description
Use first shape for bad value	This property does not apply to dynamic shapes. If the shape sequence is to be used as a status indicator, you can use the first shape to represent a 'bad value.' Set this property to **True**. This shape is only displayed when the value cannot be determine, for example, the server cannot access the associated field device.
Stylesheet	The stylesheet to use for the shape.
Description	Any notes or comments about the shape.
Title	The title of the shape file.

Related topics

"Creating a shape sequence" on page 72

Help properties

The Help properties specify 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 “Setting up Station Help” topic in the “Customizing Stations” section of the *Server and Client Configuration Guide*.)

You can specify either:

- A topic within a HTML Help file (HTML Help files have a *chm* extension.)
- An ordinary document, such as a HTML file or Microsoft Word document.

Property	Description
Topic	! **Attention** This property is only available when **HTML help** is the selected **Help type** property value. • If the **Locate topic by** property is set to **Topic file** = the HTML file name of the topic with the HTML Help file. • If the **Locate topic by** property is set to **Topic number** = the help id number associated with the topic to display.
Locate topic by	• **Topic file** = The object is always hidden • **Topic number** = The object is always visible
Help type	• **Ordinary document** = A standalone document, such as an HTML page or a Microsoft Word document. • **HTML help** = An HTML Help (**chm**) file.
Help file name	The file name of the ordinary document or HTML help file.

Keys properties

You can assign 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.

Property	Description
Key	A key and the associated action it performs when pressed.
Action	To assign an action to a key, select the key, and then select the action. Repeat for each key as required.

Lines properties

The Lines properties control the object's line thickness and style.

Property	Description
Width	The width of the line, in pixels.
Line Style	The line's style, such as dotted or dashed.
Line End Style	The style of arrow that will be displayed at the end of the line. • **No Arrow** = This option displays nothing at the beginning of the line. • **Open Arrow** = This option displays an arrow of the following style at the beginning of a line - ←. • **Block Arrow** = This option displays an arrow of the following style at the beginning of a line - ◄. • **Oval Arrow** = This option displays an arrow of the following style at the beginning of a line - ●—.
Line Begin Style	The style of arrow that will be displayed at the beginning of the line. • **No Arrow** = This option displays nothing at the beginning of the line. • **Open Arrow** = This option displays an arrow of the following style at the beginning of a line - ←. • **Block Arrow** = This option displays an arrow of the following style at the beginning of a line - ◄. • **Oval Arrow** = This option displays an arrow of the following style at the beginning of a line - ●—.
Roundness	Only applicable to a rounded rectangle. Increasing the value increases the 'roundness' of the rectangle's corners.

Page Details properties

The Page Details properties control page properties such as zooming, dynamic scanning, and security.

Property	Description
Display refresh rate in seconds	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.
Enable zooming	If this property is set to **False**, operators cannot zoom in/out on the display.
Security level	Specifies the minimum security level required by a user to call up the display.
Area type	**All assets** = All Stations can call up the display. **Only assignable asset** = Defines an assignable asset by area name. If you specify an area, only Stations with the asset assigned can call up the display. **Assignable asset defined by** = Defines an assignable asset by a file/word combination. If you specify an area, only Stations with the asset assigned can call up the display. (In general, this option is only used by Honeywell staff.)
Enable dynamic scanning	If this property is set to **True**, dynamic scanning will be triggered when the display is called up. If this property is set to **False**, dynamic scanning is disabled.

Related topics

"Periodic Task properties" on page 257

Period properties

The Period properties control the time properties of a trend or basic trend.

Property	Description
Time Period	The amount of time for which you want to see data.
Time Period Units	Select the units of time. The available options are: **Seconds**, **Minutes**, **Hours**, **Days**, **Weeks**, **Months**, **Years**.
Time Interval	The history interval.
Time Interval Units	Select the units of time. The available options are: **Section**, **Minute**, **Hour**.
Snapshot	You can choose to see snapshots or averages: **Snapshot**, **Average**.
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 in the period automatically adjusting.

Periodic Task properties

The Periodic Task properties 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.

Property	Description
Request task periodically	Select **True** to specify a periodic task.
Security Level	The security level required to perform the task.
Task LRN	The LRN number (task identifier).
Add station number offset to LRN	! **Attention** This property is deprecated and is provided only for legacy systems that have made use of it. If you set the **Add station number offset to LRN** property to **True**, 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.
Parameter 1	The first parameter for the specified LRN, if required.
Parameter 2	The second parameter for the specified LRN, if required.
Parameter 3	The third parameter for the specified LRN, if required.
Parameter 4	The fourth parameter for the specified LRN, if required.
Period (Updates)	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 **Display refresh rate in seconds** properties in the Page Details properties.

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 "Server display program" topic in the "Customizing Stations" section of the *Server and Client Configuration Guide*.
- For more information about displaying the LRN assigned to an application program, see the “Starting a task automatically” topic in the “Controlling the execution of a Server API application” section of the *Application Development Guide*.

Related topics

“Callup Task properties” on page 227

“Page Details properties” on page 255

Plot Details properties

The Plots properties specifies the plot properties within a Trend or Basic Trend display object.

Property	Description
Pen	Select the check box to make the plot visible on the trend chart.
Color	The pen color for the point parameter in the trend.
Point	The point whose values you want to display. You can click the Point Browser tab to browse points on this server or another server in the system.
Parameter	The parameter whose values you want to display.
Low Scale	The minimum value visible on the trend for the point. This value is ignored if you select the **Single scale mode** check box. Note that if you want to use the default scale (which is usually the point range), leave *both the low and high scale* at their default settings of zero.
High Scale	The maximum value visible on the trend for the point. This value is ignored if you select the **Single scale mode** check box.
Range Information	
Single scale mode	If you select this check box, the same scale is used for all points in the trend.
Y axis min	The minimum Y axis value.
Y axis max	The maximum Y axis value.

Popup properties

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

Property	Description
Initial Position	Determines the initial position of the popup when it is called up. **Determine automatically** = Station determines the initial position of the popup. **Position on parent display** = Enables you to specify the coordinates of the top-left corner of the popup. You can either type the **Left** and **Top** coordinates.
File Name	The filename of the popup display that appears when a user clicks the object.

Related topics

"Behaviors properties" on page 224

Presentation (Alphanumeric) properties

Property	Description
Display as	These properties specify how the value is displayed. The possible values are: • **Numeric** = Displays the value as a number. • **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. • **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. • **Date** = Applicable if the value is a date in Julian format. • **Date/Time** = Applicable if the value is a 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 Vendor personnel. • **Destination Address** = Applicable if the value is a destination address for a point. This option is generally only used by Vendor 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 Vendor 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 Vendor personnel. • **Duration** = Allows you to control the format when binding to a parameter of type `duration` or `deltatime`. By default, Experion presents duration data in the format D HH:MM:SS.SSS [days, hours, minutes, seconds, milliseconds]. Options are: – **Use system locale** = uses the format specified in the system's locale settings. You can choose to include milliseconds. – **D HH:MM:SS.SSS** = Hours, minutes, seconds, and milliseconds – **D HH:MM:SS**= Hours, minutes, and seconds – **D HH:MM**= Hours and minutes – **D HH** = Hours
Enable Parameter Format	When selected, displays PV parameters in the format as configured on the controller. When this option is selected, **Floating decimal point** and **Number of decimals** become inactive.
Floating decimal point	This property is available when **Numeric** or **State descriptor of point at word** is the selected **Display as** property value. Possible values are **True** or **False**.
Number of decimals	This property is available when **Numeric** or **State descriptor of point at word** is the selected **Display as** property value. The number of numbers displayed after the decimal point.
Date format	This property is available when **Date** or **Date/Time** is the selected **Display as** property value. Possible values are **Short date** or **Long date**.

Property	Description
Time format	This property is available when **Time** or **Date/Time** is the selected **Display as** property value. Possible values are **Use system locale**, **HH:MM:SS:SSSS**, **HH:MM:SS**, **HH:MM**, or **HH**.
Include milliseconds	This property is available when **Time** or **Date/Time** is the selected **Display as** property value.
Parameter name	This property is available when **Parameter of given point** is the selected **Display as** property value.
Word number	This property is available when **State descriptor of point at word** or **Parameter name of point at word** is the selected **Display as** property value.
Acronym	
Acronym Source	Possible values are **Default file** or **User file**.
File No.	This property is available when **User file** is the selected **Acronym Source** property value. The file that contains the custom acronyms.
Start at Record	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.
Start at Word	This property is available when **User file** is the selected **Acronym Source** property value. The word that contains the first acronym.
Length of Acronym	This property is available when **User file** is the selected **Acronym Source** property value. The length of the acronyms.
Location	This property is available when **User file** is the selected **Acronym Source** property value. 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.
Acronym	If you are on the server, the selected acronyms appears.
Number of characters displayed	The maximum number of characters that can be displayed. When a read-only alphanumeric is too narrow to display all of its content, an ellipsis (…) is shown to indicate that the content is truncated. There are two cases in which ellipses can be displayed: • End Ellipsis. For plain text or numeric values, ellipses appear at the end of the text. For example, Sinewave displays as 'Sinew…' • Path Ellipsis. For a path, characters in the middle of the text are replaced with ellipses. For example, 'Asset/…/Sinewave'.
Word wrap	Possible values are **True** or **False**. If **True**, the value (normally text) is displayed over several lines if it is longer than the alphanumeric's width.

Related topics

"Selecting acronyms" on page 262

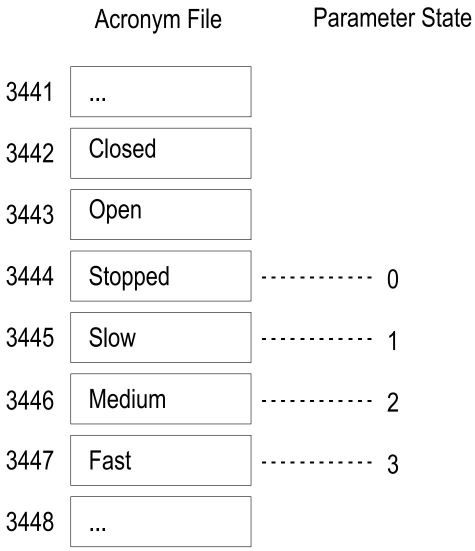
About acronyms

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.



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*.

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** as the **Display as** property value and **Default file** as the **Source** property value.
2. Complete the following properties:
   - Start at acronym
   - Number of acronyms

To select custom acronyms

1. Select **Acronym** as the **Display as** property value and **User file** as the **Source** property value.
2. Complete the following properties:
   - File
   - Start at record

- Number of acronyms
- Start at word
- Length of acronyms
- Location

Related topics

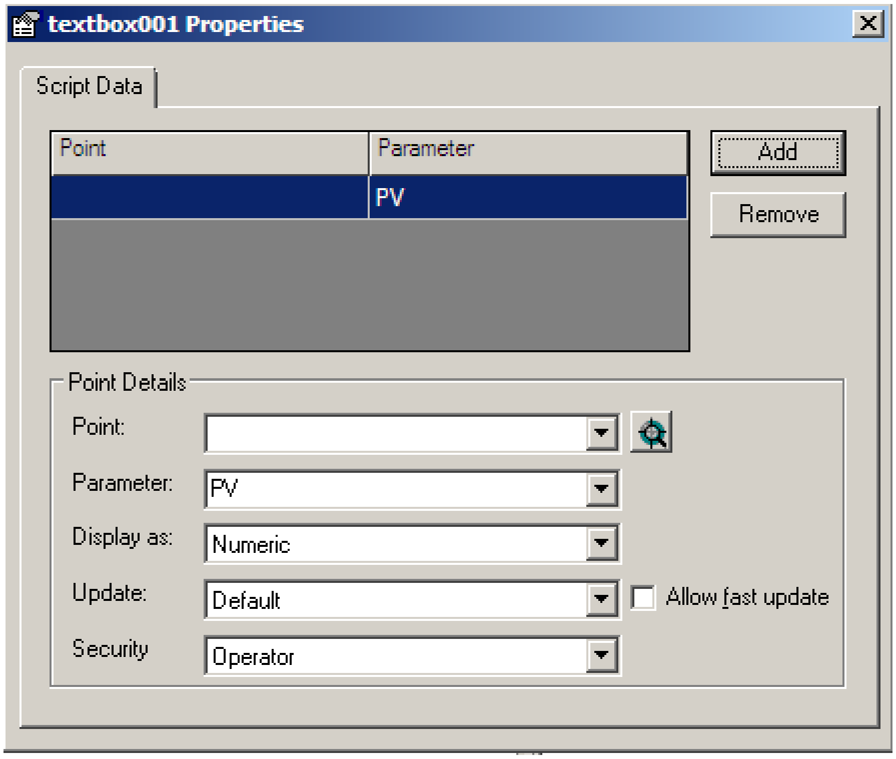
"Presentation (Alphanumeric) properties" on page 260

Presentation (Combobox) properties

Property	Description
Display as	Specifies how the object is animated. The options are: • **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. • **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 Vendor personnel.
Sort alphabetically	Alphabetically sorts the items in the list.

Script Data properties

The Script Data properties defines point parameters that can be accessed by scripts. Note that this property only appears if you select **Script Data** in the Behaviors properties.



Point:

Parameter: PV

Display as: Numeric

Update: Default

□ Allow fast update

Security Operator

Property	Description
Point	The ID of the point you want to display. If the object is in a dynamic shape, you can specify a custom property. You can click the Point Browser tab to browse points on this server or another server in the system.
Parameter	The name of the Parameter you want to display.
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**).

Property	Description
Update	The frequency that the parameter value is updated: • **Default** = The value is updated at the update rate configured for that Station. For static Stations, this rate is configured on the Main tab of the Station properties in Quick Builder. • **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**. **Attention** The actual update rate is also determined by the Station and display update rates. For more information about how the actual update rate is determined, see topic “Understanding update rates” in “Customizing Stations” in the *Server and Client Configuration Guide*
Allow fast update	Determines if the value will be updated at the fast update rate defined for that kind of Station, when the FAST key is pressed (either on the IKB or on a standard keyboard that has had a function key configured to act as a FAST key).
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.

Notes

- 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.)
- 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.)

Related topics

“Behaviors properties” on page 224

Adding point parameters to an object

To add one or more point parameters to an object

1. Select the object and then click the **Properties** tab.
2. Click on the **Script Data** property.
3. On the right-side of the **Script Data** row, click ... (ellipsis).
   The **Properties** window containing the **Script Data** tab appears.
4. Click **Add** to add a blank row.
5. Specify the point details.
6. Repeat steps 4 and 5 for any other point parameters that you want to add.

Related topics

“Using a button to start a pump” on page 575
“GetResponse3 method” on page 443

Shape Details properties

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

Property	Description
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.
Shape Name	The name of the shape file on which this shape is based. Read-only.

Shortcut properties

The Shortcut properties specify the display object's shortcut menu, which appears when a user right-clicks on the display object.

Property	Description
Shortcut file name	Only applicable if **Custom** is the selected **Shortcut** property value. Specifies the custom menu file called up by the shortcut.
Shortcut	Specifies the shortcut menu to use for this display object: • **Default** = Uses the default shortcut menu for the associated point type, if the display object is bound to a point parameter, or the default shortcut menu for the display type. For example, if the object is bound to an analog point parameter, the default shortcut menu for analog points is used. • **Custom** = Uses a specified custom shortcut menu.

Related topics

"About shortcut menus" on page 132

"About copy and paste and drag and drop behaviors" on page 208

View properties

The View properties defines Trend and Basic Trend display properties.

Property	Description
Trend title	**Attention** This property is available for Trend display objects only. The title of this trend. The title of the trend is visible only if you are also showing the Title toolbar.
Trend type	**Attention** This property is available for Trend display objects only. 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	**Attention** This property is available for Trend display objects only. The type of legend included with the trend: • **No Legend** = Legend is not 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.
Tabular history - Number of decimals	**Attention** This property is available for Trend display objects only. The number of decimals displayed in your Tabular history.
Chart type	The plot type: • **Line** • **Bar** You can only view bar chart for one point at a time. By default this is the first point on the trend.
Show plots only	**Attention** This property is available for Basic Trend display objects only. Possible values are **True** or **False**.
Show X axis	Possible values are **True** or **False**. If **False** is selected the X-axis toolbar is also hidden, even if you have selected to show the X-axis toolbar.
Show Y axis	Possible values are **True** or **False**. If **True** is selected, the Y-axis is displayed.
Show Grid lines	Possible values are **True** or **False**. If **True** is selected, grid lines are displayed.
Y axis	
Display Mode	The type of units shown on the y-axis: **Percent** or **Engineering units**.

Property	Description
In Place Editing	Possible values are **True** or **False**. If **True** is selected, a user can edit the visible range of a point using controls on the axis.
Scrollbar	Possible values are **True** or **False**. If **True** is selected, the scroll bar on the axis is visible.
Marker labels	Possible values are **True** or **False**. If **True** is selected, the scale marker labels on the axis are visible.
Axis labels	Possible values are **True** or **False**. If **True** is selected, the labels on the axis are visible.
X axis	
Show toolbar	**Attention** This property is available for Basic Trend display objects only. Possible values are **True** or **False**. If **True** is selected, the X-axis toolbar is visible.
Time selector	Possible values are **True** or **False**. If **True** is selected, the time control on the axis is visible.
Scrollbar	Possible values are **True** or **False**. If **True** is selected, the scroll bar on the axis is visible.
Marker labels	Possible values are **True** or **False**. If **True** is selected, the scale marker labels on the axis are visible.
Axis label	Possible values are **True** or **False**. If **True** is selected, the labels on the axis are visible.
Reference lines	
Width	The width, in pixels, of the reference line.
Snap to value	Causes the reference line to be placed on the nearest value.
Enable reference line	If selected, a user can add a reference line to the trend.
Show Grid lines	
Line Style	The style of the grid lines: **Solid**, **Dashed**, **Dotted**, **Dash dot**, or **Dash dot dot**
Width	The width, in pixels, of the grid lines.
Trend toolbars	
Show chart toolbar	**Attention** This property is available for Trend display objects only. Possible values are **True** or **False**. If **True** is selected, the chart toolbar is displayed.
Show x-axis toolbar	**Attention** This property is available for Trend display objects only. Possible values are **True** or **False**. If **True** is selected, the X axis toolbar is displayed.
Show title toolbar	**Attention** This property is available for Trend display objects only. Possible values are **True** or **False**. If **True** is selected, the title toolbar is displayed.

Setting up HMIWeb Display Builder

This section describes how to set up various HMIWeb Display Builder options. It also describes keyboard and menu shortcuts.

Related topics

“Setting HMIWeb Display Builder's options” on page 272
“Creating a toolbar” on page 275
“Keyboard shortcuts” on page 276
“Shortcut menu” on page 277

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” on page 272
    - “Grids and Rulers tab” on page 273
    - “Shapes tab” on page 273 (lists the folders that contain shapes)

Related topics

“Scripting languages” on page 286

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. **Attention** 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 closes unexpectedly, 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.

Registering a shape folder

You register folders that contain shapes so that the folder is available when you added the shape to a display.

To add a shape folder to the list

1. In HMIWeb Display Builder, choose **Tools** > **Options**.
2. Click the **Shapes** tab.
3. Click **Add** to add a blank line to the list.
4. Either type the folder's full path, or click the ... button and browse for the folder.
5. 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.)

Related topics

"Shape Gallery" on page 37
"Inserting a shape sequence" on page 105

Shapes tab

Option	Description
Search for shape files in	This list specifies the folders HMIWeb Display Builder searches when looking for linked shapes. 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 108.)

Option	Description
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.

Creating a toolbar

You can create your own toolbars that only contain the tools/menus you want.

To create a toolbar

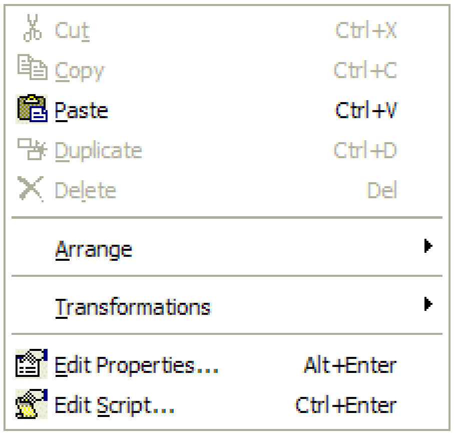
1. Choose **Tools** > **Customize**.
2. Click **New** on the Toolbars tab.
3. Type the toolbar's name and click **OK**.
   An empty—and small—toolbar appears. Its name also appears in the **Toolbars** list.
4. Click the Command tab.
5. Select a category that contains items (icons or menus) you want to 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 the above steps 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 (to the Windows printer on the computer running HMIWeb Display Builder)	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

This section 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	“Display storage techniques” on page 280
Distribution issues (replication of displays on all the servers/Stations that need them)	“Distribution (replication) issues” on page 281
Archive format (a convenient format if sending displays to remote sites)	“About archive format” on page 58

Related topics

“Display storage techniques” on page 280
“Distribution (replication) issues” on page 281

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 'Understanding and configuring file replication' section of the *Server and Client 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.) ❗ **Attention** 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, be aware that 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

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	"Scripting basics" on page 285
An introduction to the Script Editor, the tool you use to write scripts	"Using the Script Editor" on page 337
A detailed description of each object, method and event in the HMIWeb Object Model	"HMIWeb Object Model reference" on page 343
Example scenarios which include scripts that perform practical tasks	"Example scenarios" on page 545

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 scripts*—see the *Server Scripting Reference*.

Scripting basics

These topics describe basic rules and guidelines applicable to writing scripts for HMIWeb displays.

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.

Related topics

"Scripting differences between HMIWeb and DSP displays" on page 334

Scripting languages

You can write scripts in either VBScript or JScript.

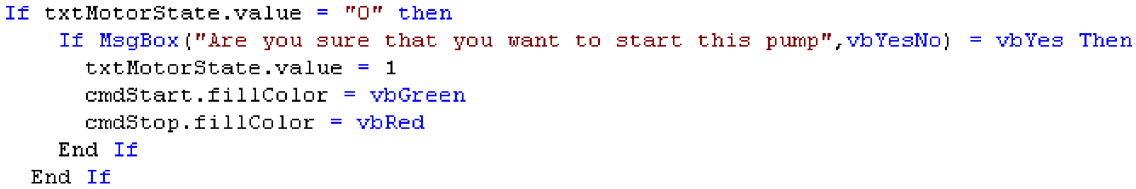
Neither scripting language requires extensive programming skills.

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



```
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
```

Figure 26: Typical script (grammar is shown in blue)

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**

Related topics

“Setting HMIWeb Display Builder's options” on page 272

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**.

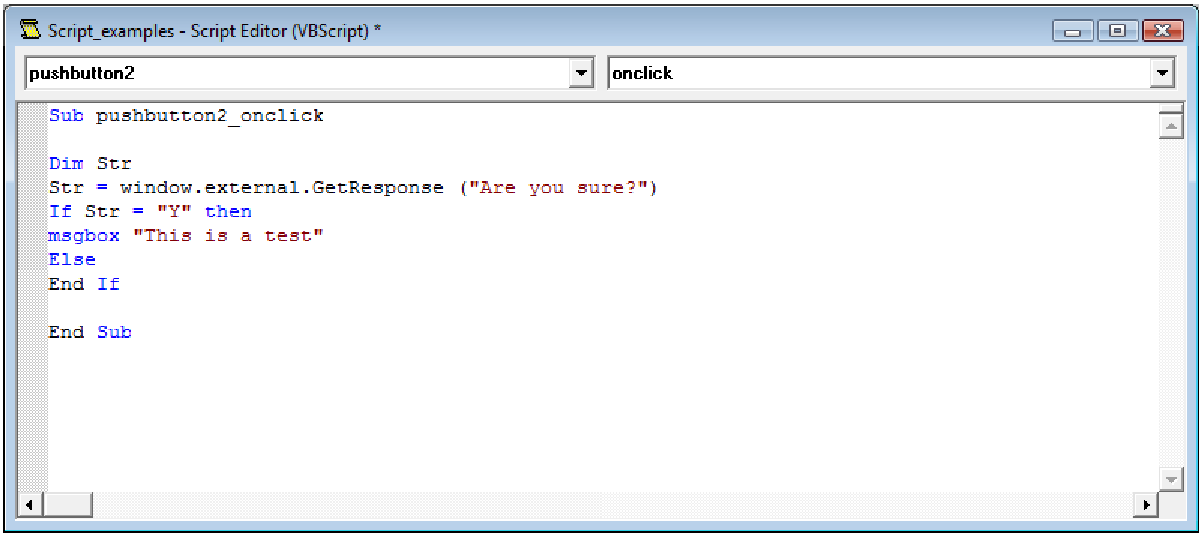
2 On the **General** tab, select the required language from **Default Scripting Language**.

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 'pushbutton002'.

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 *pushbutton002_OnClick*.



```
Sub pushbutton2_onclick

Dim Str
Str = window.external.GetResponse ("Are you sure?")
If Str = "Y" then
msgbox "This is a test"
Else
End If

End Sub
```

Tip
VBScript is case insensitive, so events written as 'onclick' or 'OnClick' or even 'ONCLICK' will all be treated the same in Station.

Event bubbling

HMIWeb displays support *event bubbling*, which means that an event passes up the object hierarchy. When an HTML element generates an event, the event handler for that element first handles it. If the event is not canceled by the element event handler, the event passes to the parent element for handling. The event continues up ("bubbles up") the object hierarchy until it is canceled, or until it reaches the root object, the document object. 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.

Attention
Not all HTML objects support event bubbling. The event's description indicates if it does not bubble.

It is possible that a single event is handled several times, in some way at each level in the object hierarchy, however, this is not necessary. Event bubbling allows you to create more efficient event handling by allowing 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 display. Such a script would check which object fired the event and then perform the appropriate task.

Event bubbling is useful because it:

- Allows multiple common actions to be handled centrally
- Reduces the amount of overall code in the display
- Reduces the number of code changes required to update a display

Event bubbling canceling

All objects which support event bubbling, bubble to their parent element (and, recursively, all the way up to the document object) unless the event bubbling is canceled. To cancel event bubbling, you must set the *window.event.cancelBubble* property to '*true*' in the event handler. Unless canceled, events will bubble up the hierarchy and be handled by all parent elements of the object firing the event, even if the event has already been handled.

You can use the Event object—which represents the current event—to prevent event bubbling at any level in the object hierarchy by setting the event cancelBubble property to *true* at the appropriate level event handler.

For example, the following statement prevents event bubbling past an object called 'group2'

```
<SCRIPT language=VBScript event=onclick for=group2 defer>
    on Error resume next
    'other onclick action script goes here

    window.event.cancelBubble = true
</SCRIPT>
```

Canceling event bubbling should not be confused with canceling the event or canceling the action for the event. Canceling an event's bubbling does not cancel the event or cancel its action.

- The Object Explorer shows the display's object hierarchy.
- Not all events bubble through the object hierarchy. For example, OnClick, OnActivate and OnDeactivate bubble but OnFocus and OnBlur do not. The event's description indicates if it does not bubble.

Event canceling

Returning '*false*' from an event handler, or setting *window.event.returnValue* to '*false*', cancels the event.

- Not all events can be canceled. The event description states whether it cannot be canceled. For example, the OnActivate and OnDeactivate events cannot be canceled, nor can the OnMouseMove, OnMouseEnter, OnMouseLeave, and OnMouseOut events.
- Canceling an event does not cancel the event bubbling. If you want to cancel event bubbling, it must be canceled separately.
- Canceling the event does not cancel the data change event action if the display element is data-bound to the server database and displaying a data value being changed by the operator. If you want to cancel the data change event action in this situation, it must be canceled in the OnChange event using the CancelChange method.

Event action canceling

In the situation where the display element is data-bound to the server database and displaying a data value being changed by the operator, you can use the CancelChange method only in the OnChange event to cancel the data change event action before it is sent to the server.

- Canceling the data change event action does not cancel the event. If you want to cancel the event, it must be canceled separately.
- The CancelChange method is only applicable to data-bound display elements which permit operator change in value, including the Alphanumeric object, Checkbox object, Combobox object, or Pushbutton object.

Related topics

“Event object” on page 367

“Page object” on page 378

“Objects method” on page 459

“Application object” on page 353

“Using event bubbling to reduce script effort” on page 577

Event firing order

HMIWeb display events fire in an order dependent upon which elements are used and the order in which actions (events) occur on the display. Certain actions will cause specific events to fire, so the exact firing order of events on a display with user interaction is not necessarily predictable. The following table lists some common events and the actions that trigger them.

Event	Trigger
OnActivate	Occurs every time the display is loaded, reloaded or refreshed. Also occurs every time an element becomes the active element in the display. This typically occurs when input focus is moved between elements in a display. Once an element is active it will remain active until another element becomes the active element. The active element can lose focus and remain active. For example when an AlphaNumeric object is clicked in a display it becomes the active element. If the operator then clicks off the display, perhaps on something like the Station toolbar, the AlphaNumeric object has lost focus yet still remains active. If the operator clicks on the AlphaNumeric object again, it regains focus but does not fire the OnActivate event because it was already the active element. Note also that the Page object will receive all bubbling OnActivate events from every element on the display, unless the event bubbling has been specifically canceled in the event handler for the element. Also note that the OnActivate event action itself cannot be canceled, even though its bubbling can be canceled. The OnActivate event of the element gaining the activation is preceded by the OnDeactivate event of the element losing activation, if one is currently activated. If an element has been clicked or keyboard navigated to, the OnActivate event is the precursor to the OnBlur event of the element losing the focus, and to the OnFocus event of the element gaining the focus. Compare with OnDeactivate and OnFocus events.
OnBlur	Occurs when an element loses the current input focus, for example, when the keyboard navigation is used or the mouse is clicked on the display background or on another element, or on another display or application. Does not bubble. The OnBlur event of the element losing the focus is often immediately preceded by the OnActivate event of the element gaining the focus. The OnBlur event of the element losing the focus is usually the immediate precursor to the OnFocus event of the element gaining the focus. Compare with OnFocus and OnActivate events.
OnClick	Occurs when the mouse is clicked whilst positioned over an HTML object such as the display or an element. The basic order of mouse click events are: OnMouseOver > OnMouseDown > OnMouseUp > OnClick. Note that other events may occur within the above order of basic mouse click events, such as the OnMouseEnter and OnMouseMove events, the OnDeactivate and OnBlur events of the element losing input focus, and the subsequent OnActivate and OnFocus events of the element being clicked upon. Note also that the Page object will receive all bubbling OnClick events from every element on the display, unless the event bubbling has been specifically canceled in the event handler for the element. Compare with OnDblClick event.

Event	Trigger
OnDblClick	Occurs when the mouse is double-clicked whilst positioned over an HTML object such as the display or an element. Note that the double-click is determined by the Windows mouse property settings, using the time period between clicks, and the screen position co-ordinates of the mouse pointer for the first and second clicks. The basic order of events are: OnMouseOver > OnMouseDown > OnMouseUp > OnClick > OnMouseUp > OnDblClick Note that the OnClick event (and all the others listed in the order of events above) will fire every time before the OnDblClick event, so all actions associated with these events, if any, are executed before the OnDblClick event actions are executed. Also note that the second OnMouseDown and OnClick events do not fire, as might be logically expected in the order of events. This is the behavior of Windows. Note that the Page object will receive all bubbling OnDblClick events from every element on the display, unless the event bubbling has been specifically canceled in the event handler for the element. Compare with OnClick event.
OnDeactivate	Occurs every time the active element on the display becomes de-active, that is, whenever another element on the same display becomes active. Note that an element can lose focus yet remain the active element, and therefore will not fire the OnDeactivate event until another element on the same display becomes the active element, or until the display is closed (unloaded). Also note that the OnDeactivate event action itself cannot be canceled, even though its bubbling can be canceled. Compare with OnActivate and OnBlur events.
OnFocus	Occurs when a display or display element receives input focus. Usually occurs through mouse click or keyboard navigation, however can also occur through script. Note that the onActivate event usually fires before the OnFocus event. Does not bubble. Compare with OnBlur and OnActivate events.
OnKeyDown	Occurs when a keyboard key is depressed. If the key is held down, this event along with the OnKeyPress event will repeat in pairs at the rate determined by the Windows key repeat value as set on the Station computer. Note that the display element with the current focus will receive the keystroke events. This is the precursor to the OnKeyPress event. Compare with OnKeyPress and OnMouseDown events.
OnKeyPress	Occurs when an alphanumeric key is pressed on the keyboard. If the key is held down, this event along with the OnKeyDown event will repeat in pairs at the rate determined by the Windows key repeat value as set on the Station computer. Note that the display element with the current focus will receive the keystroke events. The order of events are: OnKeyDown > OnKeyPress > OnKeyUp This is the precursor to the OnKeyUp event. Compare with OnKeyDown and OnKeyUp events.
OnKeyUp	Occurs when a keyboard key is released. Note that the display element with the current focus will receive the keystroke events. Compare with OnKeyPress and OnMouseUp events.

Event	Trigger
OnMouseDown	Occurs when the mouse button is depressed on a display element. Note that this action brings the activation and input focus to the element being clicked, which fires the OnDeactivate and OnBlur events for the element losing the input focus, and then fires the OnActivate and OnFocus events for the element gaining the focus. Can be canceled, and does bubble. Compare with OnMouseUp event.
OnMouseEnter	Occurs when the mouse enters a display element. Typically, this event is used in conjunction with the OnMouseLeave event to indicate (in the message zone) which element the mouse pointer is hovering over. Note that the onMouseOver event usually fires before the OnMouseEnter event. Also note that the OnMouseEnter event action itself cannot be canceled, and that the event does not bubble. Compare with OnMouseLeave event.
OnMouseLeave	Occurs when the mouse leaves a display element. Typically, this event is used in conjunction with the OnMouseEnter event to clear (in the message zone) which element the mouse pointer is hovering over. Note that the OnMouseOut event usually fires before the OnMouseLeave event. Also note that the OnMouseLeave event action itself cannot be canceled, and that the event does not bubble. Compare with OnMouseEnter event.
OnMouseMove	Occurs when the mouse moves on a display element, for example when the mouse first makes contact with a display element before the OnMouseOver and OnMouseEnter events, and repeatedly whilst the mouse moves over the element. Note that the OnMouseMove event action itself cannot be canceled. However, unlike the OnMouseEnter and OnMouseLeave events, note that the OnMouseMove event does bubble.
OnMouseOut	Occurs when the mouse leaves an element. This is usually the precursor to the OnMouseLeave event. Note that the OnMouseOut event action itself cannot be canceled. However, unlike the OnMouseLeave event, the OnMouseOut event does bubble. Compare with OnMouseOver event.
OnMouseOver	Occurs when the mouse enters an element. This is usually the precursor to the OnMouseEnter event. Note however, that unlike the OnMouseEnter event, the OnMouseOver event can be canceled, and does bubble. Compare with OnMouseOut event.
OnMouseUp	Occurs when the mouse button is released. This is the precursor to the OnClick and OnDblClick events. Can be canceled, and does bubble. Compare with OnMouseDown and onClick events.
OnOperatorKey	Occurs when an operator key is pressed on the Integrated Keyboard (IKB).

Event	Trigger
OnPageComplete	Occurs after the display has completely loaded, and all display elements have been updated. The display loading sequence of events are: OnActivate > OnFocus > OnUpdate (individually of all display elements that are data-bound to the server database) > OnPageComplete Other display events may still occur even after this event has fired. Note that the display will receive all element events that bubble and have not been stopped by the element event handler. Note that the OnPageComplete event should be used in preference to the OnLoad event because the OnLoad event fires before all display elements have been updated, whereas the OnPageComplete event fires after all display elements have been loaded and updated with data. Compare with DHTML OnLoad event. ! **Attention** OnPageComplete does not fire on Pan and Zoom displays. Instead, refer to the OnViewportChanged event.
OnShapeLoad	Occurs when the content of the shape is available, but before the elements within the shape have received their initial data update.
OnUnload	Occurs when one display is changed to another one. Note that the currently active element on the display may fire its OnDeactivate event before the display OnUnload event.
OnViewportChanged	Occurs when all the elements in a view have received a data update and can be scripted against. Fires multiple times – whenever the viewport is changed and some data bound elements come into view.

Normally, you would expect OnActivate to fire before OnPageComplete, but in most cases, OnPageComplete usually fires first because no elements on the display have been activated yet. However, if you have a script that activates an object before the display fully loads, then OnActivate would fire first.

Scripts that are meant to run when the display has finished loading should only ever be placed in the OnPageComplete event handler, and never be placed in the OnActivate event or the DHTML OnLoad event handler of the display, since some elements may not have finished loading when the display OnActivate event is fired, and it may fire every time the display gains focus.

Sequence of event examples

Page initial load event sequence

The sequence of events for a page initial load include:

1. <server data-bound element> OnUpdate event (one per element)
2. Page OnPageComplete event

Note that the page has not been given activation and focus, unlike the reload action event sequences as described further below.

Page close (unload) event sequence

The sequence of events for a page closure include:

1. Page OnDeactivate event (if the page window is active)
2. Page OnUnload event

Note that the OnDeactivate event only occurs if the page window is active and had focus before being closed.

Page reload (refresh) with focus event sequence

The sequence of events for a page reload (if page already has focus) include:

1. Page OnDeactivate event
2. Page OnUnload event
3. Page OnActivate event
4. Page OnFocus event
5. <server data-bound element> OnUpdate event (one per element)
6. Page OnPageComplete event

Note that the OnDeactivate and OnUnload events occur before the page is reloaded, and that after reloading, the page is given activation and focus, unlike the initial page load event sequence described previously above.

Page reload (refresh) without focus event sequence

The sequence of events for a page reload (if page does not have focus) include:

1. Page OnActivate event
2. Page OnFocus event
3. Page OnDeactivate event
4. Page OnUnload event
5. Page OnActivate event
6. Page OnFocus event
7. <server data-bound element> OnUpdate event (one per element)
8. Page OnPageComplete event

Note that the page is first given activation and focus before being reloaded, and that after reloading the page is again given activation and focus.

Mouse over and off an element event sequence (with no clicking)

The sequence of events for passing the mouse across an element include:

1. Element OnMouseMove event
2. Element OnMouseOver event
3. Element OnMouseEnter event
4. Element OnMouseMove event (may occur repeatedly whilst over element)
5. Element OnMouseOut event
6. Element OnMouseLeave event

Note that the OnMouseOver and OnMouseOut events are paired, as are the OnMouseEnter and OnMouseLeave events paired to each other.

Note also that the OnMouseOver event usually fires before the OnMouseEnter event, and if so, the OnMouseOut event will fire before the OnMouseLeave event. However, it has been observed that this pairing sequence occurs on average 8 times out of 10. For the other times, they still occur as ordered pairs, however, the OnMouseEnter event occasionally fires before the OnMouseOver event, and if so, the OnMouseLeave event will fire before the OnMouseOut event.

Mouse over and click an element (with focus) event sequence

The sequence of events for clicking an element which already has the current focus include:

1. Element OnMouseMove event
2. Element OnMouseOver event
3. Element OnMouseEnter event
4. Element OnMouseMove event (may occur repeatedly whilst over element)

5. Element OnMouseDown event
6. Element OnMouseUp event
7. Element OnClick event

Mouse over and click an element (without focus) event sequence

For the purposes of this example, there are two elements named Element1 and Element2. Element1 has the current focus and Element2 is about to be clicked.

The sequence of events for clicking an element which currently does not have focus include:

1. Element2 OnMouseMove event
2. Element2 OnMouseOver event
3. Element2 OnMouseEnter event
4. Element2 OnMouseMove event (may occur repeatedly whilst over element)
5. Element2 OnMouseDown event
6. Element1 OnDeactivate event (this event occurred on Element 1)
7. Element2 OnActivate event
8. Element1 OnBlur event (this event occurred on Element 1)
9. Element2 OnFocus event
10. Element2 OnMouseUp event
11. Element2 OnClick event

Note that if either of these elements are server data-bound, the OnUpdate event for those data-bound elements will be fired as the focus moves from element to element. This is standard display behavior of Station to update display elements.

Viewing events

You can temporarily view events by sending them to the Station message zone or a VBScript MessageBox. Alternatively, you can store and view messages at a later time by sending messages to the HMIWeb log file.

To view the HMIWeb Log file, launch the Honeywell Log Viewer (HLV) tool located at **Start** > **All Programs** > **Honeywell Experion PKS** > **Server** > **Diagnostic Tools** > **Experion PKS Server Log**., and click the '**log.txt [Client]**' tab.

To create script that sends a message (such as an event occurrence) to the Station message zone, VBS MessageBox, or HMIWeb log file, open the display (that you want to capture events from) in HMIWeb Display Builder, open the VBScript editor and copy the following script to the page **[General]** section:

```
option explicit
on error resume next
dim tracer 'event tracer
dim tracker 'event tracker
dim msg 'message string
dim separator 'formatting dot and space separator = ". " ' set full stop and following space
character
tracer = 0 'initialise

Sub LogEvent(evnt)
    'increment tracer
     tracer = tracer + 1

    'create message
    msg = "Event tracking item #" & tracer & separator & evnt

    'show event in station message zone
     window.external.MessageZoneText = msg

    'show event tracker messagebox
    tracker = "Event Tracker" & vbnewline & "Created: " & now
    msgbox tracker & vbnewline & msg

    'log event
    window.external.LogMessage msg
End sub
```

You do not need to perform all three message display methods. Comment out the methods you do not require. (You may quickly find that the VBScript MessageBox is inappropriate for multiple messages.)

To track events, insert one line of script into the event handler for each event of interest. For example, if you wanted to track the OnUpdate event for a server data-bound alphanumeric display element named *AN1*, you need to select *AN1* in the VBScript editor object box, *onupdate* in the event box, and enter the following example script:

```
Sub AN1_onupdate
  LogEvent("AN1 onupdate")
End Sub
```

Save the script (and display), call up the display in Station, and observe that the event appears or is logged as appropriate to your script.

Similarly, if you want to capture the Enter and Leave mouse events for a particular display element such as an alphanumeric named *AN1*, you add the appropriate single line of script to those event handlers as in the following example:

```
Sub AN1_onmouseenter
  LogEvent("AN1 onmouseenter")
End Sub

Sub AN1_onmouseleave
  LogEvent("AN1 onmouseleave")
End Sub
```

Typically, these events are used together to indicate (in the message zone) which element the mouse pointer is hovering over.

Save the script (and display), call up the display in Station, and observe that those events appear or are logged as appropriate to your script, whenever the mouse enters or leaves the *AN1* display element.

Related topics

"Displaying messages to operators" on page 298
"OnActivate event" on page 497
"OnBlur event" on page 501
"OnClick event" on page 504
"OnDblClick event" on page 508
"OnDeactivate event" on page 508
"OnFocus event" on page 512
"OnKeyDown event" on page 513
"OnKeyPress event" on page 513
"OnKeyUp event" on page 514
"OnMouseDown event" on page 516
"OnMouseEnter event" on page 517
"OnMouseLeave event" on page 517
"OnMouseMove event" on page 518
"OnMouseOut event" on page 519
"OnMouseOver event" on page 519
"OnMouseUp event" on page 520
"OnOperatorKey event" on page 523
"OnPageComplete event" on page 527
"OnUnload event" on page 538

Event interception and handling

When handling events, be aware of the different intended purposes for seemingly similar events. For example, the OnMouseOver and OnMouseOut events are complementary, and each respectively precedes the complementary OnMouseEnter and OnMouseLeave events. The differences between these pairs of events is to do with their timing order and behavior.

The OnMouseOver event usually occurs first, can be canceled, and bubbles in the event hierarchy, whereas the OnMouseEnter event usually occurs second, cannot be canceled and does not bubble. Likewise, their respective corresponding events behave in a similar manner, where the OnMouseOut event usually occurs before the OnMouseLeave event, the OnMouseOut event bubbles whereas the OnMouseLeave event does not bubble, and neither event can be canceled.

The OnMouseEnter and OnMouseLeave events are ideal for performing actions that do not need to be canceled, or do not need to bubble to a common event handler. Typically, the OnMouseEnter event is used in conjunction with the OnMouseLeave event to indicate (in the message zone) which element the mouse pointer is hovering over.

Alternatively, the OnMouseOver and OnMouseOut events are suitable for actions that may be common to multiple display elements on the same page, such as a group of options or buttons or labels or shapes, because they permit event bubbling which allows for common page level event handling.

Different events can occur by seemingly similar actions. With an alphanumeric element for example, the OnChange event occurs when the object's value (or some other property) is changed by the operator. However, the OnUpdate event occurs whenever the value is changed by the server or a script.

If the value was changed by the operator (selecting the alphanumeric element on the display page and entering keyboard characters), the OnChange event (for the alphanumeric) is fired before the changed value is sent to the server database. This allows for the OnChange event handler to intercept the value, validate it, manipulate it, and either pass it on to the server or cancel the change.

Alternatively, if the value was changed in the field, or by script, or in the server database, by whatever cause, the onUpdate event for the alphanumeric display element is fired only after the value has been updated to the display from the server.

General section

The *general section* is typically used for setting default script statements, declaring global variables, storing general-purpose functions and subroutines that are called by other scripts, and for storing common error handlers.

Any scripts in the general section are executed as soon as the display loads.

All variables declared outside of subroutines or functions in the general section are global in scope, and available to any script of any display element on the page. This is useful for counters and flags.

The *`option explicit`* is a VBScript statement which instructs the script engine to throw an error whenever it encounters an undeclared variable. This is good practice when writing and developing your scripts to ensure that you are using the proper variable as expected, and are not accidentally using some similarly named or different scope variable. The option explicit statement, if used, must be placed first in the general section:

```
option explicit
```

The default behavior of the script engine is to stop running the script whenever an unhandled error is encountered at runtime. During script development and testing, such behavior may be unhelpful. To prevent your scripts from stopping upon error encounters, use the *`on error resume next`* VBScript statement in the general section. This will cause the script engine to continue processing and resume interpreting the next line in the script after the line that caused the error to occur. When placed in the general section, this statement has global scope for all script used on the page:

```
on error resume next
```

Declaring and initializing VBScript variables in the general section makes those variables global in scope and available to all scripts used on the page:

```
dim counter 'tracking counter
counter = 0 'initialize
dim isReady 'boolean status flag
isReady = false 'initialize
```

Related topics

"Using HMIWeb error handlers" on page 333

"Creating a moving object" on page 560

"Creating an animation with a shape sequence" on page 562

Displaying messages to operators

For many applications other than Experion, you would use a message box to display information to the operator. For example, if you were using VBScript, you would typically use the MsgBox function. However, such message boxes are *modal* to the window that launched them, which means they prevent the operator from interacting with the display until the message box is acknowledged. (Something that could be a critical issue during process upset conditions.)

Instead of using message boxes, Honeywell strongly recommends that you use temporary message zone text or callouts to display messages. Refer to the topic "Station object" for additional information about displaying temporary messages using window.external.TemporaryMessageZoneText. Refer to "showCallout method" for details about callouts.

Note that a script-generated message will be overwritten by a server-generated message because the latter is deemed to be more important.

If the primary reason you want to display a message is to help debug a script, you should use `window.external.logmessage` to write the message to the HMIWeb log file, `<data folder>\Honeywell\HMIWebLog\`.

Where `<data folder>` is the location where Experion data is stored. For default installations, `<data folder>` is `C:\ProgramData`. The `C:\ProgramData` folder is a system folder, which means that it is only visible if you select the **Show hidden files, folders, and drives** option button in the **Folder Options** dialog box. To change this setting in Windows Explorer, click **Organize** > **Folder and search options**, and then click the **View** tab.

Related topics

“Station object” on page 392

“Application object” on page 353

“Event firing order” on page 289

“showCallout method” on page 493

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. The topic “Creating a moving object” 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 200 ms 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 canceled 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” on page 562

Related topics

"Creating a moving object" on page 560
"Creating an animation with a shape sequence" on page 562

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.

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 or 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.application
```

The following example shows how to call a subroutine, 'MyRoutine,' in the parent display from a popup:

```
window.external.parentDocument.parentWindow.MyRoutine(value)
```

Related topics

“Event object” on page 367
“Page object” on page 378
“Objects method” on page 459
“Application object” on page 353

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.

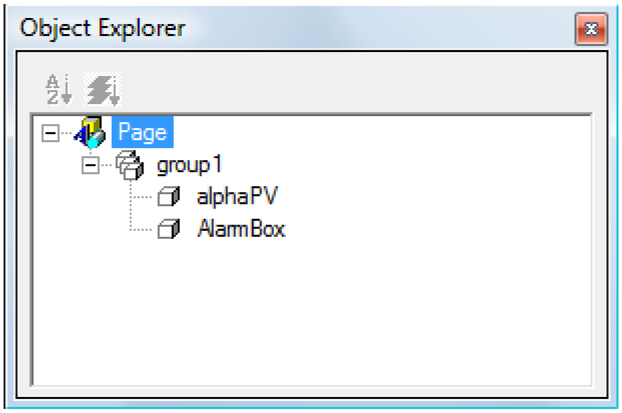
The performance improvements are particularly significant if there are multiple instances of the shape in a display. (This occurs because the functions are 'loaded' into the general section of the display and are called by all instances of the shape.)

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 Object 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
```

Related topics

"Shape object" on page 390

"Objects method" on page 459

Working with popups and faceplates

Scripting in a multi-window environment

When you are working in a multi-window environment the only faceplate scripting property that is supported is:

```
window.external.parentDocument
```

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 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.application
```

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
```

Related topics

“Application object” on page 353

“Station object” on page 392

Accessing Station-related properties in page-level scripts

Page-level scripts (scripts in the current display) 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 Bar (a RuntimeStatus object property).

```
x = window.external.Parent.RuntimeStatus.Time
```

Related topics

"Station object" on page 392

"AppWindow object" on page 359

"RuntimeStatus object" on page 388

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 = 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://msdn2.microsoft.com/en-us/library/ms531197(VS.85).aspx.
- 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:

  See “VBScript color constants”.

  ```
  rect002.fillColor = vbBlue
  ```
- 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.

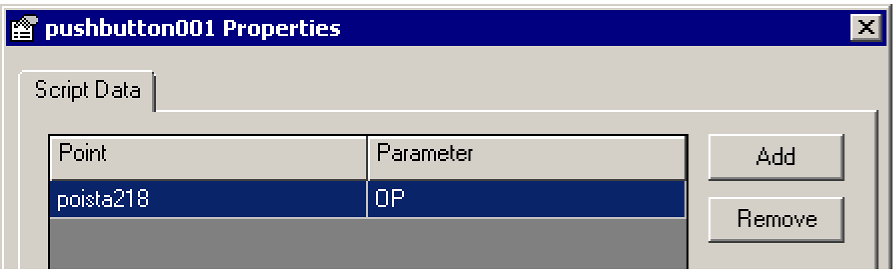
Adding a point parameter to the Script Data tab of a display object also adds a corresponding set of point-related properties and methods of the Script Data behavior to the display object.

Notes

- Adding a point parameter to the Script Data tab of one display object makes the parameter accessible to all scripts in the display.
- When adding Script Data behavior, you can select any object within the scope of the behavior, but it is generally more convenient for debugging and maintenance purposes to select the object to which you want to attach your scripting.
- The point-related properties of the Script Data behavior are automatically updated whenever there is a change in a parameter value of the point.

Examples

These examples show how to read and write to the parameter shown in the following figure.



pushbutton001 Properties

Script Data

Point	Parameter
poista218	OP

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
```

If appropriate you can write to the parameter using a security level that is higher than the current operator. For example, set the value with ENGR (4) security level:

```
pushbutton001.SetDataValueWithSecurityLevel "poista218.OP", 1, 4
```

Example scenarios

“Using a button to start a pump” on page 575

“Creating a jogging control” on page 559

Related topics

"Properties and methods of the Script Data behavior" on page 309
"OnDataChange event" on page 507
"Using a button to start a pump" on page 575
"RequestDemandData method" on page 465
"Using color to reflect the value of a status point" on page 576
"Creating a jogging control" on page 559

Properties and methods of the Script Data behavior

These properties and methods of the Script Data behavior only exist for an object if one or more point parameters are listed on the display object's Script Data tab.

Property/method	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 either: • 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" on page 465 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*.
DataQuality(point.parameter)	Returns the data quality of the specified point.
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*.
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*.

Property/method	Description
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.parameter)	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 this 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.
SetDataValueSynchronouslyWithSecurityLevel(point.parameter, value, SecurityLevel)	Synchronously sets the specified point parameter to the specified value, using the specified security level. ('Synchronously sets' means that the script waits until the value has been written to the field device/controller before continuing.) The *SecurityLevel* can be: *0* = LVL1 (View only) *1* = LVL2 (Acknowledge only) *2* = OPER *3* = SUPV *4* = ENGR *5* = MNGR Returns *True* if the write is successful, otherwise returns *False*. Note that this 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. Note that the MNGR level can only be used if the **Allow controls with "MNGR" security level from Station scripts** option is enabled. If using SecurityLevel *LVL1*, *LVL2*, *OPER*, *SUPV*, or *ENGR*, the currently logged in Operator or Station must have full access to the asset to which the point being written to belongs. If using SecurityLevel *MNGR*, any point in the system can be written to regardless of the scope of responsibility of the currently logged in Operator or Station.

Property/method	Description
SetDataValueWithSecurityLevel(point.parameter, value, SecurityLevel)	Sets the specified point parameter to the specified value, using the specified security level. The *SecurityLevel* can be: *0* = LVL1 (View only) *1* = LVL2 (Acknowledge only) *2* = OPER *3* = SUPV *4* = ENGR *5* = MNGR Returns *True* if the write is successful, otherwise returns *False*. Note that the MNGR level can only be used if the **Allow controls with "MNGR" security level from Station scripts** option is enabled. If using SecurityLevel *LVL1*, *LVL2*, *OPER*, *SUPV*, or *ENGR*, the currently logged in Operator or Station must have full access to the asset to which the point being written to belongs. If using SecurityLevel *MNGR*, any point in the system can be written to regardless of the scope of responsibility of the currently logged in Operator or Station.
UnacknowledgedAlarm(Point.Parameter)	Returns *True* if the specified point has unacknowledged alarms, otherwise returns *False*.

Related topics

"Reading and writing to point parameters" on page 308

"Accessing the point mapped to a shape custom property" on page 316

Using custom properties

Custom properties specific to shapes (with a scope restricted to only within the shape) are referred to as '*shape custom properties*'. Shape custom properties are configured in HMIWeb Display Builder on the **Custom Properties** tab in the **Shape Properties Window**.

Shape custom properties are often used to store user-defined settings common to the display objects within the shape, such as fillcolor and linewidth.

Custom properties specific to the display page (with a scope of 'Display') are referred to as '*system custom properties*'. System custom properties are configured in HMIWeb Display Builder on the **Custom Properties** tab in the **Display Properties Window**.

System custom properties with 'Display' scope are often used to store user-defined settings which are common to several display objects within the display page, in the same manner that shape custom properties are used to store common display properties for several objects within a shape.

Custom properties specific to Station (with a scope of 'Station') are also referred to as '*system custom properties*'. System custom properties are configured in HMIWeb Display Builder on the **Custom Properties** tab in the **Display Properties Window**.

System custom properties with 'Station' scope are often used with generic displays which are customized using script when the display is called up in Station.

However, the real power of a system custom property comes into play when the system custom properties are used as an intermediary reference to points—that is, the custom property is dynamically mapped to a point. This allows for the mapping to the actual point to be set or changed at runtime using script, whilst keeping the display object reference to the custom property unchanged. The display object statically references the custom property, and the custom property dynamically references (is mapped to) the actual point.

This use of the system custom property means that once setup, you do not ever have to change the display object point reference, as it is referring to the point name stored in the custom property, and not referring to the actual point itself.

Format of the alarm status

The DataAlarm property returns the alarm status of the specified point parameter. The format of the alarm status is as follows:

Empty (12 bits) Severity (10 bits)

												H	K	E	L	P	S	Q	C	A	R										

Item	Description
H	The alert shelving condition: 0 = Alert is not shelved 1 = Alert is shelved
K	The alert acknowledgement condition: 0 = Alert is not acknowledged 1 = Alert is acknowledged
E	The alert acknowledgement requirement: 0 = Alert acknowledgement is not required 1 = Alert acknowledgement is required
L	The alert condition: 0 = Alert condition is off (that is, no alert) 1 = Alert condition is on (that is, in alert)
P	The alarm suppression condition: 0 = Alarm is not suppressed 1 = Alarm is suppressed
S	The alarm shelving condition: 0 = Alarm is not shelved 1 = Alarm is shelved
Q	The alarm quality: 0 = Alarm condition calculation is valid 1 = Alarm condition calculation is invalid (for example bad limit value)
C	The alarm condition: 0 = Alarm condition is off (that is, no alarm) 1 = Alarm condition is on (that is, in alarm)
A	The acknowledgement status: 0 = Alarm is not acknowledged 1 = Alarm is acknowledged
R	The acknowledgement requirement: 0 = Acknowledgement is not required 1 = Acknowledgement is required

Item	Description
Severity	The alarm severity, a value between 1 and 1000.

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 that accesses points is operating correctly, you should (during development) use *window.external.logmessage* to write the point values to the HMIWeb log file: *<data folder>\Honeywell\HMIWebLog\*.

Where *<data folder>* is the location where Experion data is stored. For default installations, *<data folder>* is *C:\ProgramData*. The *C:\ProgramData* folder is a system folder, which means that it is only visible if you select the **Show hidden files, folders, and drives** option button in the **Folder Options** dialog box. To change this setting in Windows Explorer, click **Organize** > **Folder and search options**, and then click the **View** tab.

Using custom properties in scripts

Scripts can use custom properties in many ways, for example:

- If a custom property is mapped to a point, scripts can:
  - Access the point
  - Re-map the custom property to another point at run time
- If a custom property is not mapped to a point, scripts can use it to store user-defined values.

Related topics

"Mapping a point to a custom property using script" on page 314
"Accessing the point mapped to a shape custom property" on page 316
"Example: Getting the values of the parameters mapped to shape custom properties" on page 316
"Accessing the point mapped to a system custom property" on page 317
"Example: Getting the values of the parameters mapped to system custom properties" on page 317
"Setting the value of a custom property when calling up a display or faceplate" on page 319
"Example: Re-mapping a shape custom property to another point at run time" on page 319
"Example: Re-mapping a system custom property to another point at run time" on page 320
"Example: Using a shape custom property to store a user-defined value" on page 321
"Example: Using a system custom property to store a user-defined value" on page 321

Mapping a point to a custom property using script

After a custom property has been declared in a shape or in a display or in Station, it can be referenced by other display elements within its scope.

For example, a shape custom property can be accessed from any script within the shape, whereas a system custom property with the scope of 'Display' can be accessed from any script on the same display, and a system custom property with the scope of 'Station' can be accessed from any script on any display in the Station, as well as from Station Scripting Objects (SSO).

Shape custom properties are available to scripting only whilst the display containing the shape is loaded.

System custom properties with the scope of 'Display' are stored in the display data repository and are also only available whilst the display is loaded.

System custom properties with the scope of 'Station' are stored in the Station data repository and are available for the life of the Station.

The steps to map a point to a custom property at runtime using script involve:

1. Calling the *PutValue* method of the data repository object appropriate to the scope of the custom property.
2. Passing the custom property name and point name in the method arguments.

For example, you have declared a system custom property with a scope of 'Display' and named it *PointName*. You also have an actual point named *Pt12345* that you want to map to the custom property and have its value used in the display. In your script for an appropriate event handler, insert the following line:

```
DisplayDataRepository.PutValue "PointName", "Pt12345"
```

When run, this maps the actual point named *Pt12345* to the display system custom property named *PointName*, which can be used by any display element object on the display that uses that Script Data behavior custom property as its point.

Attention

Do not attempt to use the value of a custom property (and therefore its mapped point) immediately after mapping a custom property to a point.

The underlying connection between the custom property and the mapped point uses a subscription service to subscribe the custom property to the point. Consequently, there is a delay from when the point is subscribed to when the value comes back to the mapped custom property from the server.

To separate the actions of mapping points to custom properties from setting (or reading) the point value, it is necessary to separate those functions. The recommended solution is to place the script that accesses the mapped point value into the *onupdate* event handler of the display object. The *onupdate* event only fires when the value of the display element object is updated from the server, so you can be sure that the value is correct and up-to-date.

You need to calculate the volumetric flow rate for a pipe segment in your plant, and show that value in an alphanumeric display element adjacent to the visual representation of the pipe segment in an HMIWeb display.

The rate is determined from the current flow velocity value multiplied by the known flow area of the pipe. In this example, you have a point (named *PS1958FV*) in your project, which measures the flow velocity (in pipe segment #1958). The pipe segment internal diameter is 4 inches.

From these values, you can utilize fluid mechanics to determine the current volumetric flow rate using the continuity equation formula solving for flow rate 'Q=Av', where:Q=flow rate; A=flow area; and v=flow velocity.

You have a pushbutton (named *PBPS1958CV*) on the display adjacent to the pipe segment controlling the activation of the pipe segment using a flow control valve. Press the button and the control valve opens, activating the pipe segment.

A separate alphanumeric display element (named *ANPS1958FV*) is used to show the flow velocity within the pipe segment (in inches per second). This item is bound to the custom property named *FlowRatePoint* in the Script Data tab of the alphanumeric's properties. The *FlowRatePoint* custom property is mapped to the actual point in this example (named *PS1958FV*) as described above.

When the pipe is activated, and its contents are flowing, the velocity value is updated from the server and sent to the custom property, which in turn is bound to the alphanumeric. Each time the alphanumeric is updated, it fires its *onupdate* event, which in this example is used to calculate the volumetric flow rate of the pipe segment (in cubic feet per minute). This calculated flow rate value is shown in the alphanumeric named *ANPS1958FR*.

Notice how the mapping of the point to the custom property occurred in a separate script to the reading and subsequent use of the point value.

```
[General section]
option explicit 'explicitly declare all names
on error resume next 'continue even if in error
'create global variable
Dim PipeSizeIn
Dim ConvertFactor

Sub PBPS1958CV_onclick
  'assign (map) points to custom properties
  DisplayDataRepository.PutValue "FlowRatePoint", "PS1958FV"
  'declare other values used in the flow rate calculation
  ConvertFactor = 5 'multiply by 60 and divide by 12
  PipeSizeIn = 4 'this variable could be passed
  'in other ways, however, in order to simplify and focus
  'this example, it has been hardcoded instead.

Sub ANPS1958FV_onupdate
  'flow velocity value has just been updated from the server
  'create local variable
  Dim FlowRate
  'call function to make use of updated value
  'and calculate flow rate of pipe segment
  FlowRate = CalcFlowRate(me.value, PipeSizeIn)
```

```
    'convert result flow rate from
    'inches per second to feet per minute
    'and assign resultant value to Flow Rate AN
    ANPS1958FR.value = FlowRate * ConvertFactor
  End Sub

  Function CalcFlowRate(vel, dia)
    Dim Rad, Pi, csa 'create variables
    Rad = dia/2 'radius is half diameter
    Pi = 4*ATN(1) 'Pi is 4 times the arctangent of 1
    csa = Pi*Rad*Rad 'cross sectional area is Pi times R squared
    CalcFlowRate = vel*csa 'flow rate is velocity times area
  End Sub
```

Accessing the point mapped to a shape custom property

You can write scripts that use a shape object's custom properties to access whichever points are mapped to those custom properties.

Notes

- In order to use custom properties in scripts, you must first add them to an object's behavior **Script Data** tab. You can select any object, but from a system maintenance viewpoint, it is generally more convenient to select the object to which you want to attach your scripting.
- You define the common shape custom properties in the **Custom Properties** tab of the **Shape Properties Window**, and you reference those common shape custom properties from any display object in the shape, through the **Script Data** tab of the display object's behavior. The shape custom properties are mapped to actual points at runtime using script (not shown in the following example).

Related topics

“Properties and methods of the Script Data behavior” on page 309

“OnDataChange event” on page 507

“Using a button to start a pump” on page 575

“Example: Getting the values of the parameters mapped to system custom properties” on page 317

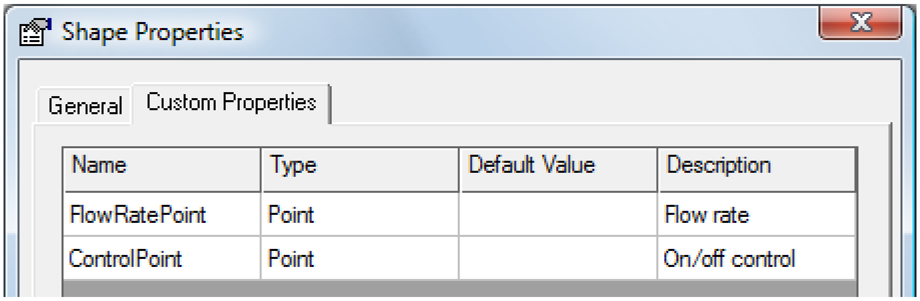
Example: Getting the values of the parameters mapped to shape custom properties

This example uses the ondatachange event script to demonstrate how to obtain the values of the point parameters mapped to the object’s behavior custom properties “ControlPoint” and “FlowRatePoint”.

To create the shape and write the script

1 In HMIWeb Display Builder, on a display, create the shape.

2 Specify the shape's custom properties.

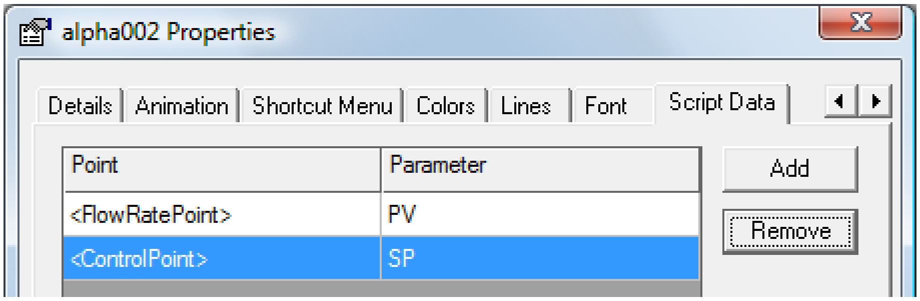


Name	Type	Default Value	Description
FlowRatePoint	Point		Flow rate
ControlPoint	Point		On/off control

3 Add display objects to the shape.

4 Assign the object behavior custom properties to the dynamic objects. Select an appropriate shape object (in this example, “alpha002”). In the **Behaviors** tab of the **Properties Window**, select **Script Data**.

5 Click the Script Data tab and add your custom properties. (The angle brackets indicate that a “point” is a custom property, which gets evaluated at runtime for the actual point name stored in the custom property.)



Point	Parameter
<FlowRatePoint>	PV
<ControlPoint>	SP

6 Open the Script Editor and write the following script.

```
Sub alpha002_ondatachange
  'Obtain the point parameter values
  PumpStatus = alpha002.DataValue("ControlPoint.SP")

'Not used in this example. Just provided to demonstrate
'how to access the actual point parameter at runtime
'using the DataValue method of the ScriptData behavior

Liters = alpha002.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 those custom properties.

Notes

- In order to use custom properties in scripts, you must first add them to an object's behavior **Script Data** tab. You can select any object, but from a system maintenance viewpoint, it is generally more convenient to select the object to which you want to attach your scripting.
- You define the common system custom properties in the **Custom Properties** tab of the **Display Properties Window**, and you reference those common system custom properties from any display object in the display, through the **Script Data** tab of the display object's behavior. The system custom properties are mapped to actual points at runtime using script (not shown in the following example).

Related topics

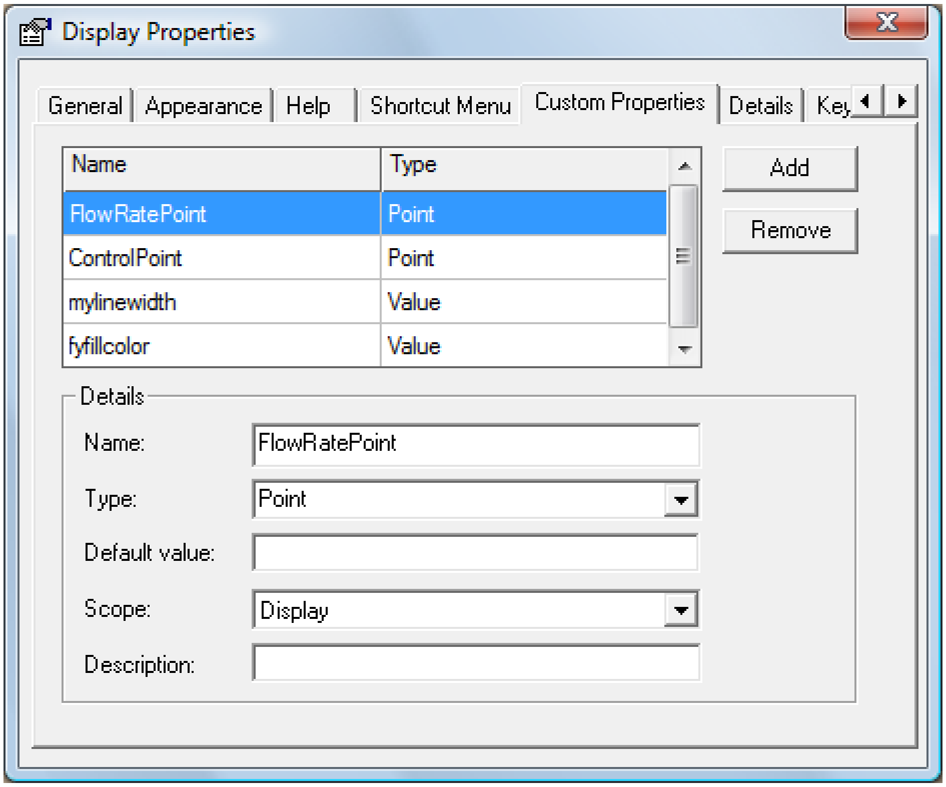
“Changing a trend's contents at run time - Scenario 1” on page 548

Example: Getting the values of the parameters mapped to system custom properties

This example uses the ondatachange event script to demonstrate how to obtain the values of the point parameters mapped to the object's behavior custom properties 'ControlPoint' and 'FlowRatePoint'.

To insert the object and write the script

1. In HMIWeb Display Builder, open or create the display .
2. Specify the display's custom properties. In this instance, in the **Display Properties Window**, click the **Add** button, enter the name 'FlowRatePoint' and again click the **Add** button, enter the name 'ControlPoint', and close the Display Properties Window. (The 'Point' type and 'Display' scope are selected and applied by default.)



Name	Type
FlowRatePoint	Point
ControlPoint	Point
mylinewidth	Value
fyfillcolor	Value

Name: FlowRatePoint

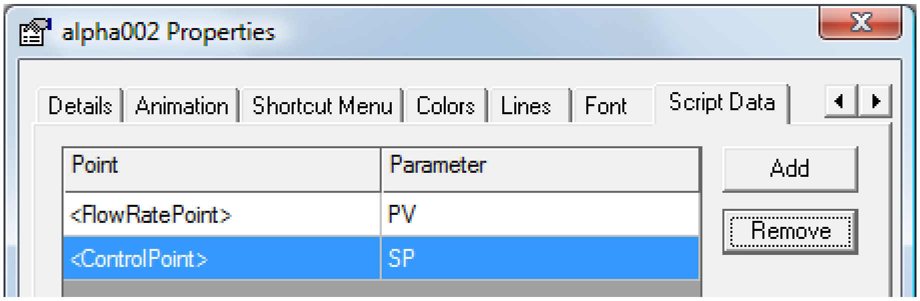
Type: Point

Default value:

Scope: Display

Description:

3. Assign the custom properties to an appropriate object (in this example, 'alpha002'). In the Behaviors tab of the Properties Window, select **Script Data**—this adds the Script Data tab.
4. Click the **Script Data** tab and add the custom properties. (The angle brackets indicate that a 'point' is a custom property, which gets evaluated at runtime for the actual point name stored in the custom property.)



Point	Parameter
<FlowRatePoint>	PV
<ControlPoint>	SP

5. Open the Script Editor and write the following script.

```
Sub alpha002_ondatachange
  'Obtain the point parameter values
  PumpStatus = alpha002.DataValue("ControlPoint.SP")

  'Not used in this example. Just provided to demonstrate
  'how to access the actual point parameter at runtime
  'using the DataValue method of the Script Data behavior
  Liters = alpha002.DataValue("FlowRatePoint.PV")

  'Now check the pump's status
```

```
        If PumpStatus = 1 then
          'Do something
        Else
          'Do something else
        End If
    End Sub
```

Related topics

“Accessing the point mapped to a shape custom property” on page 316

Setting the value of a custom property when calling up a display or faceplate

Setting the value of a custom property when calling up a display

You can set the value of a system custom property in a display (such as a generic display) when you call up the display using the following syntax:

displayname?custompropertyname=value

where:

- *displayname* is the name of the display
- *custompropertyname* is the name of the custom property
- *value* is the value to which the custom property is set

Specifying the point ID when calling up a faceplate

You can specify the point name for a faceplate when you call it. This example uses the CreatePopupWindow2 method to call up the faceplate for an analog point called 'Tank1'. (*sysdtlana_fp.htm* is the Experion faceplate for a standard analog points.)

```
HDXPopupBehavior.CreatePopupWindow2 "sysdtlana_fp.htm?currentpoint=tank1",0, 0, 1, True
```

Example: Re-mapping a shape custom property to another point at run time

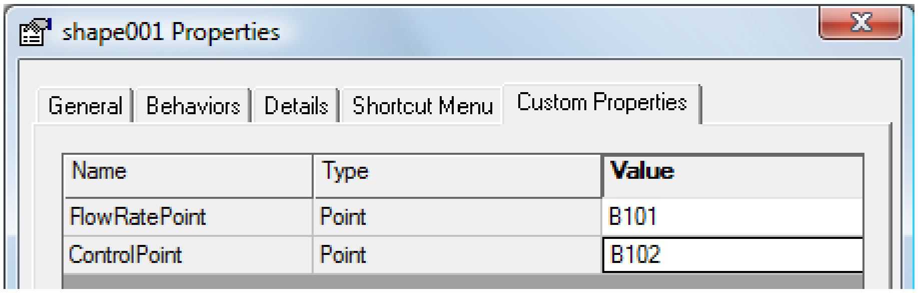
When you insert a dynamic shape into a display, you map each custom property to a particular point. However, you can write scripts that re-map the custom properties to other points at run time.

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.)



Name	Type	Value
FlowRatePoint	Point	B101
ControlPoint	Point	B102

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
```

Example: 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.

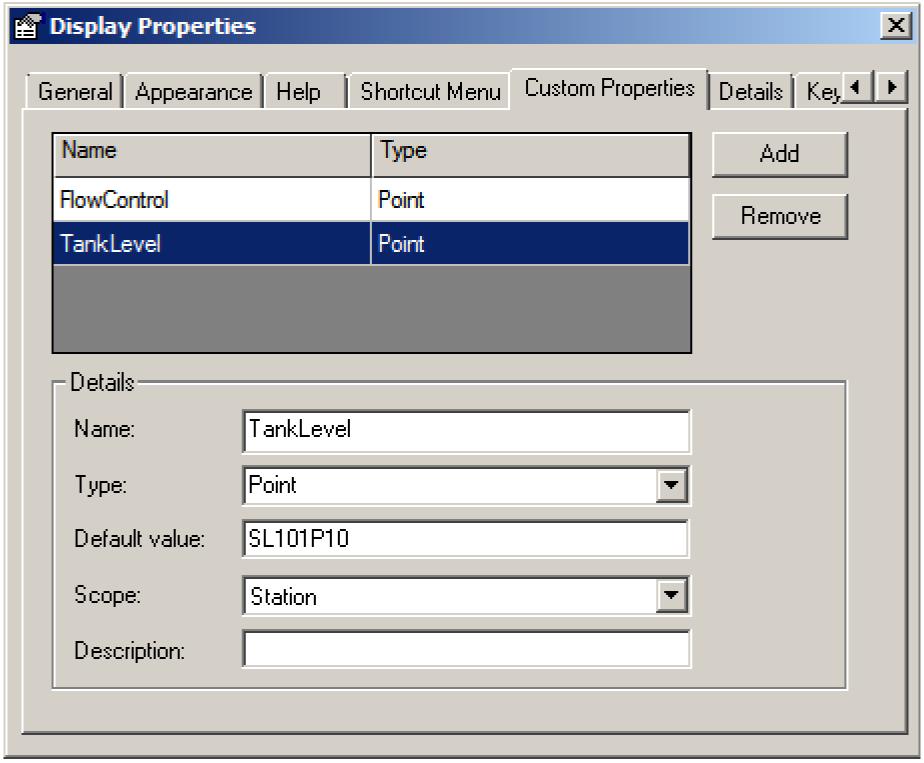
This example shows how to write a script for the onclick event of a pushbutton, which re-maps the system custom properties to two other points. The scope of the system custom properties has been set to *Station*. (In this example, the custom properties are initially mapped to 'FIC100' and 'SL101P10' 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.)



Name	Type
FlowControl	Point
TankLevel	Point

Name: TankLevel

Type: Point

Default value: SL101P10

Scope: Station

Description:

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"
  window.external.StationDataRepository.PutValue"TankLevel", "SL101P20"
  window.external.StationDataRepository.autoReloadContent = True
End Sub
```

Example: 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):

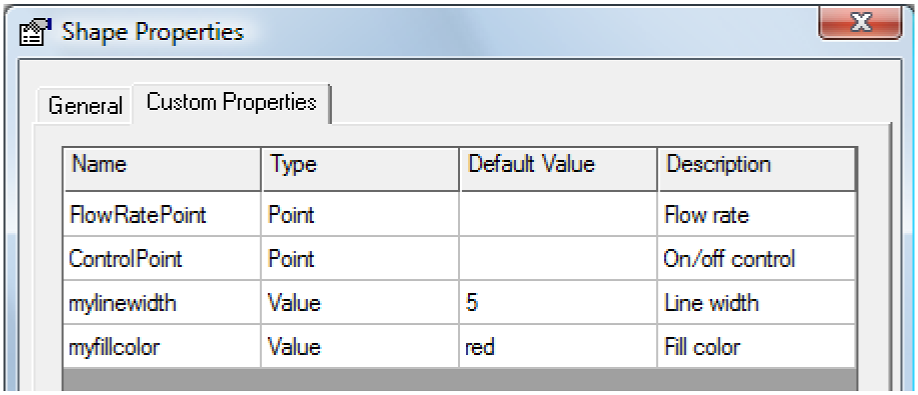
  {%*custompropertytype*::*custompropertyname*%}
- Use the SetCustomProperty method to change the value of custom property.

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.

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*.



Name	Type	Default Value	Description
FlowRatePoint	Point		Flow rate
ControlPoint	Point		On/off control
mylinewidth	Value	5	Line width
myfillcolor	Value	red	Fill color

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
```

Related topics

"GetCustomProperty method" on page 430

"SetCustomProperty method" on page 472

"GetValue method" on page 447

"PutValue method" on page 460

Example: 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.

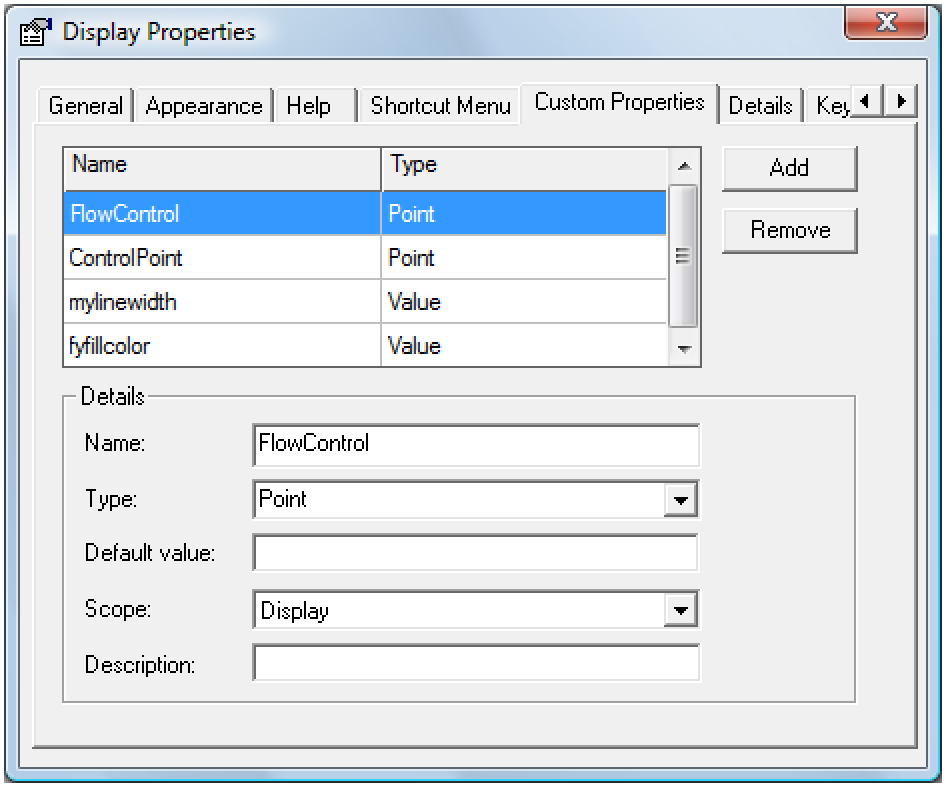
You cannot use {%*custompropertytype*::*custompropertyname*%} for system custom properties.

This example shows how to write an onclick script for a rectangle ('rect001') that uses custom properties to set its line width.

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`.



Name	Type
FlowControl	Point
ControlPoint	Point
mylinewidth	Value
fyfillcolor	Value

Name: FlowControl

Type: Point

Default value:

Scope: Display

Description:

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"
End Sub
```

Related topics

“GetValue method” on page 447

“PutValue method” on page 460

Finding the name of a Station command

Some methods and events return or invoke Station commands. You can find out the names of the commands through Station's connection properties.

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

If you need to know the name of a command associated with a shortcut key and you use specialized keyboards, such as the IKB, you also need to use the *GetActionIDForCommand* method to obtain the key's ActionID. (This is necessary because the dedicated keys on these keyboards have specific command names for their keys. For example, the 'Select Setpoint' command is 'OKB_SP' for the SP key on an IKB.)

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**.

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.

Related topics

"StationWindows object" on page 395

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.

Configuring Custom Events in Station

To enable custom events, you must configure Station to permit use of the Desktop Framework Event Model (DFEM) event system.

To enable custom events in Station

1. Open the **Connection Properties** dialog, go to the **Scripting** tab
2. In the **Station Scripting Objects** section, click the **Add** button.
3. Edit the new entry so that it reads:

   ```
   ProgID=DFEMDirector.DFEMList
   ```
4. Click **Save**.

Related topics

"RaiseCustomEvent method" on page 461

"OnCustomEvent event" on page 505

Scripting trends

Because a trend object (and to a lesser extent a basic trend object) is such a complex object, you can significantly increase its usefulness by writing scripts for it—for example, to add another point to the trend, or to hide/show gridlines.

Attention

The following script examples are for simple demonstration purposes only. In particular, it is recommended that you avoid using message boxes in on-process graphics. Best practice is to use the message zone or callouts for user input and notifications.

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
```

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

```
Sub pushbutton001_onclick
basictrend001.chartArea.Plots.SetBoundPlot 1, "Sinewave",
"pv"
End Sub
```

Changing trend views

Attention

Trend views are not available for basic trends.

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

Attention

Legends are not available for basic trends.

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

> **Attention**
> Toolbars are not available for basic trends.

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

> **Attention**
> Trend titles are not available for basic trends.

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 average:

```
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 in a trend.

```
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
```

The following script will change the plot to blue in a basic trend.

```
Sub pushbutton010_onclick
basictrend001.chartArea.Plots(1).color = vbBlue
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 trend type

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

```
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 481.

```
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
```

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.

Related topics

"Scripting differences between HMIWeb and DSP displays" on page 334
"Style object" on page 396
"CancelChange method" on page 413
"CancelEvent method" on page 414
"Using the Script Editor in Station" on page 338
"Application object" on page 353

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 located in *<install folder>\Honeywell\Experion PKS\Client\Station\Samples*, where *<install folder>* is the location where Experion is installed.

Related topics

"OnConnect event" on page 504

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, Honeywell recommends 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 HMIWeb 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.

Prototype VBScript

```
sub MyErrorHandlerFunction
  ...
end sub

function MyErrorHandlerFunction
  ...
  MyErrorHandlerFunction = true
end function
```

Prototype JScript

```
function MyErrorHandlerFunction()
{
  ...
  return true;
}
```

Related topics

“General section” on page 297

“Page object” on page 378

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.

Dictionary object

DSP scripts access the Dictionary object through the Application object, for example:

```
Application.Dictionary.Add "Volts", 12.2
```

HMIWeb scripts, however, access the Dictionary object through Window.External, for example:

```
Windows.External.Dictionary.Add "Volts", 12.2
```

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 and CancelEvent methods on dynamic objects (objects that are linked to the database).

Related topics

“Scripting basics” on page 285
“Writing Station-level scripts” on page 330

DHTML Reference

Microsoft Web site

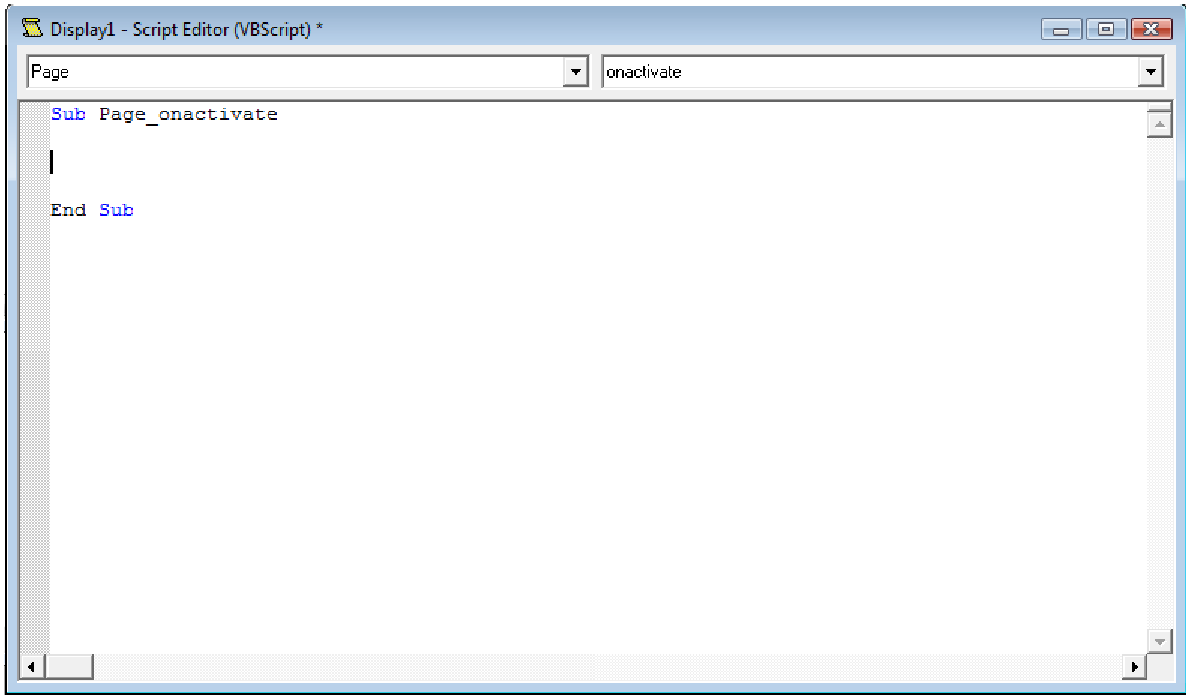
Microsoft provides comprehensive documentation for the DOM and DHTML at its Web site. At the time of writing, the address is http://msdn.microsoft.com/en-us/library/ms533050(VS.85).aspx.

Using the Script Editor

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.)



```
Sub Page_onactivate


End Sub
```

Using the 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.

Note that Station-based scripts should not attempt to access the Application object until the OnAppStartup event has fired.

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.

Related topics

"Writing Station-level scripts" on page 330

"Application object" on page 353

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

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**.

To write a script

1 Click to open the Script Editor.

> **Attention**
>
> 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.

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.

Finding or replacing strings in scripts

1. Choose **Edit** > **Find** to search the display's scripts for instances of a specified string.
2. If you want to replace one string with another, choose **Edit** > **Replace**.

HMIWeb Object Model reference

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.

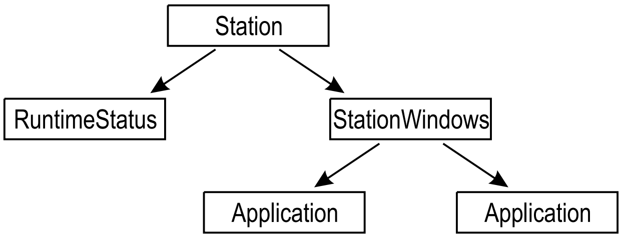
For a detailed description of the DHTML object model, go to Microsoft's MSDN Web site at http://msdn.microsoft.com/en-us/library/ms533050(VS.85).aspx.

Related topics

“Overview of the HMIWeb Object Model” on page 344
“Overview of the Trend and BasicTrend object models” on page 345
“Objects” on page 347
“Methods” on page 406
“Events” on page 496

Overview of the HMIWeb Object Model

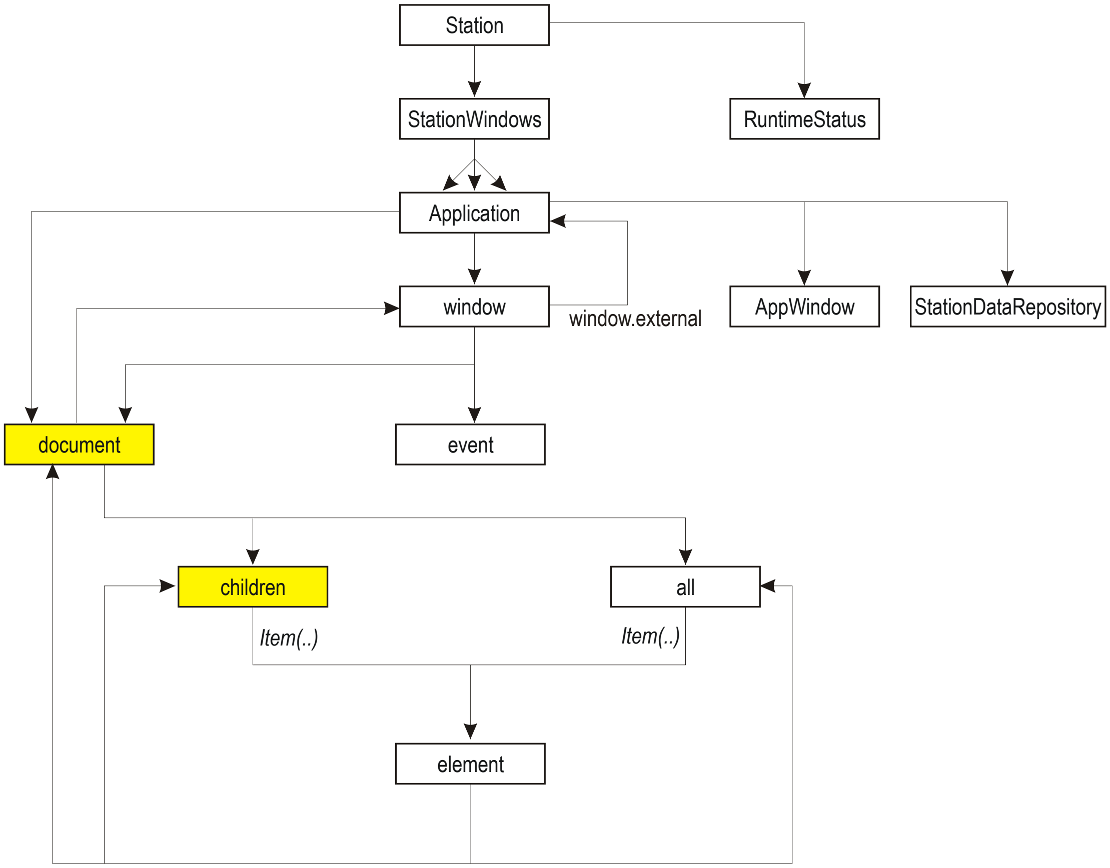
The following figure shows the levels of the HMIWeb Object Model. The top level is the Station object, which contains Station-wide settings, such as title bar format, timers, and so on. From Station object you can access the RuntimeStatus and StationWindows properties. The RuntimeStatus object contains 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.



Attention

Scripting on the alarm and event tables is not supported.

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).



Overview of the Trend and BasicTrend object models

The following figures show the Trend and BasicTrend object models.

The Trend object is a complex object that exposes each major part of the trend. The BasicTrend object model, however, lacks the extra child objects that provide advanced scripting capabilities.

Note that this legacy object is only provided for backward compatibility. It exposes the Chart object, which was used in earlier versions of HMIWeb Display Builder. (If you have a display that uses the Chart object, Honeywell strongly recommends that you migrate it.)

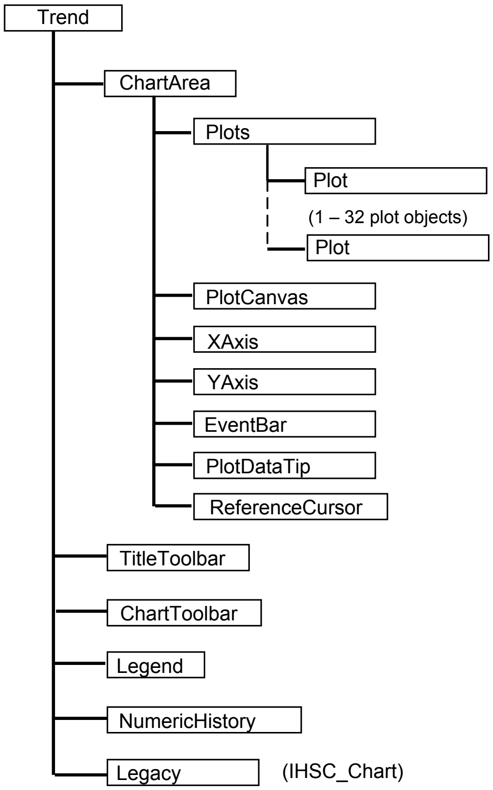


Figure 27: Trend object model

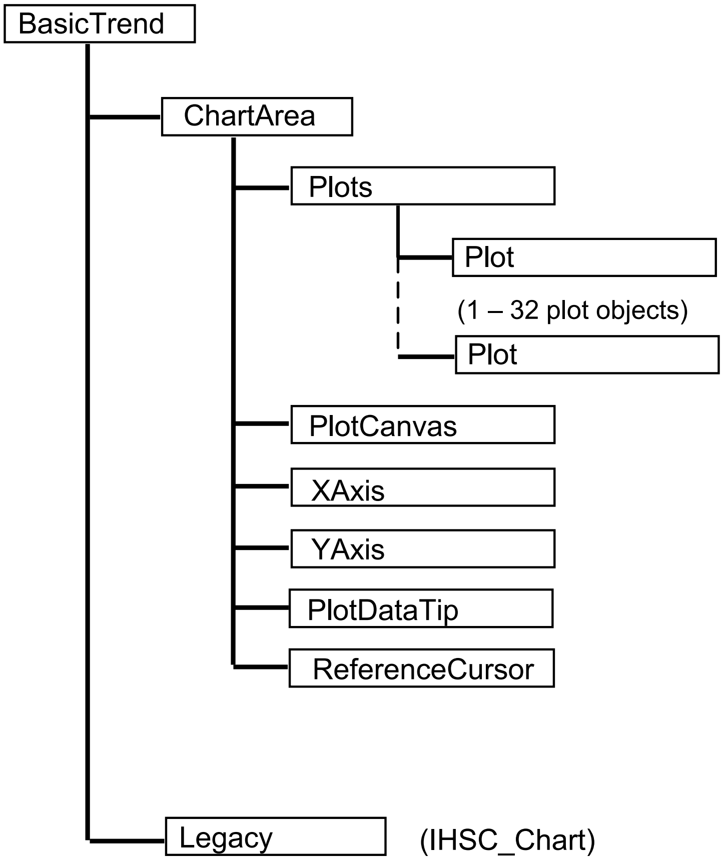


Figure 28: BasicTrend object model

Objects

The objects are listed in alphabetical order.

Related topics

“ActiveX control” on page 348
“ActiveX document object” on page 348
“AlarmState object” on page 348
“Alphanumeric object” on page 350
“Application object” on page 353
“AppWindow object” on page 359
“BasicTrend object” on page 360
“ChartArea object” on page 361
“ChartToolbar object” on page 361
“Checkbox object” on page 362
“Combobox object” on page 363
“Dictionary object” on page 365
“DisplayDataRepository object” on page 366
“Event object” on page 367
“EventBar object” on page 368
“Group object” on page 369
“Hyperlink object” on page 370
“Indicator object” on page 372
“Legacy object” on page 374
“Legend object” on page 375
“Node object” on page 376
“Nodes object” on page 377
“NumericHistory object” on page 377
“Page object” on page 378
“Picture object” on page 380
“Plot object” on page 381
“Plots object” on page 382
“PlotCanvas object” on page 382
“PlotDataTip object” on page 383
“Popup object” on page 385
“Pushbutton object” on page 386
“ReferenceCursor object” on page 387
“RuntimeStatus object” on page 388
“Shape object” on page 390
“Station object” on page 392
“StationDataRepository object” on page 394
“StationWindows object” on page 395
“Style object” on page 396
“Text object” on page 397
“TitleToolbar object” on page 398
“Trend object” on page 399
“Vector Graphic object” on page 400
“window object” on page 402

"XAxis object" on page 403

"YAxis object" on page 404

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 operators.

ActiveX document 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.
disabled	Returns or sets operator control of the object, the value is either *True* or *False*.
file	Returns or sets the name (and path) of the document.
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.
style	Returns or sets the object's style-related properties.

Methods

None.

Events

"OnDataChange event" on page 507

AlarmState object

Description

The AlarmState object represents the alarm state icon, which is typically used to show the alarm state of an associated point. However, if you do not link the icon to a point, you can use scripts to set the icon to a particular alarm state.

Properties

Property	Description
Acknowledged	Sets the acknowledged state of the Alarm state icon image, which can be: • True = Acknowledged • False = Unacknowledged

Property	Description
Priority	Sets the priority of the Alarm state icon image, which can be: • 3 = Urgent • 2 = High • 1 = Low • 0 = No alarm
Rtn	Sets the Active State of the Alarm state icon image, which can be: • True = Return to normal • False = Active

Methods

None.

Events

"OnActivate event" on page 497

"OnBlur event" on page 501

"OnClick event" on page 504

"OnContextMenu event" on page 505

"OnDblClick event" on page 508

"OnDeactivate event" on page 508

"OnFocus event" on page 512

"OnKeyDown event" on page 513

"OnKeyPress event" on page 513

"OnKeyUp event" on page 514

"OnMouseDown event" on page 516

"OnMouseEnter event" on page 517

"OnMouseLeave event" on page 517

"OnMouseMove event" on page 518

"OnMouseOut event" on page 519

"OnMouseOver event" on page 519

"OnMouseUp event" on page 520

Alphanumeric object

Properties

Property	Description
currentValue	Returns the current value of the object as last (previously) set by the server or script. **Attention** The currentValue property is designed to be used in the OnUpdate and OnChange events to check what the previous value was (for validation or comparison purposes before the update process has completed) even after the value property has been updated with the new value. The currentValue property isn't updated from the value property until immediately after the OnUpdate or OnChange event has completed.
*Data** *Quality** *AlarmSeverity*	These properties only exist if one or more point parameters are listed on the object's **Script Data** tab.
disabled	Returns or sets operator control of the object, the value is either *True* or *False*.
ElementINFState	Returns whether a value is infinity or not. Returns *0* for non-infinity values, *-1* for negative infinity values or *1* for positive infinity values.
fillColor	Returns or sets the object's fill color.
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.
IsElementNaN	Returns *True* if the value of the element is not a number (NaN), otherwise, returns *False*.
lineColor	Returns or sets the object's line color.
lineColorBlink	Returns or sets the blinking state of the object's line color. The value is either *True* or *False*.
NumericDisplayFormat	Sets how a numeric value is displayed in an AlphaNumeric object. See the remarks section for the correct syntax for this property. Use caution when setting this property as incorrect formats may cause undefined results. Modifying this parameter can also have consequences for the object's properties on the **Details** tab.
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.
styleClass	Returns or sets the object's style.
textColor	Returns or sets the object's text color. The color can be either the RGB value (24-bit) or one of the predefined colors.
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. **Attention** If you set the value of an alphanumeric linked to the database, the value will not change until the server has acknowledged the change. The *value* property is also reflected in the *currentValue* property after an update has been completed. During an update initiated by the server or through a script, the value property is immediately updated, and the OnUpdate event is triggered. During this time, the *currentValue* property retains the previously set value until the OnUpdate event has completed.

Remarks

- The syntax of the *NumericDisplayFormat* property is:

  “%[flags][width][.precision] type “

  where *type* can be:

  - *d*, print as a signed decimal number.
  - *u*, print as an unsigned decimal number.
  - *f*, print a double in normal (fixed-point) notation.
  - *g*, print a double in either normal or exponential notation, whichever is more appropriate for its magnitude. This type differs slightly from fixed-point notation in that insignificant zeroes to the right of the decimal point are not included. Also, the decimal point is not included on whole numbers.
  - *x* or *X*, print an unsigned in as a hexadecimal number.*x* uses lower-case letters and *X* uses upper-case letters.
  - *o*, print an unsigned int in octal.
  - *s,* print a character string.

  where *flags* can be:

  - *+*, the output will always show the sign + or - of a number (the default is to omit the sign for positive numbers).
  - *-*, the output is left-aligned to the placeholder (the default is to right-align the output).
  - *#*, alternate form. For type *g*, trailing zeroes are not removed. For type *f*, the output always contains a decimal point.
  - A space will include leading spaces in the output. If combined with 0 it causes the sign to become a space when positive, but the remaining characters will be zero-padded.
  - *0*, leading zeroes are included (instead of spaces) to fill a fixed length field. For example if value is 3 and format is *“%2d”* the result is “ *3*”, while the format *“%02d”* results in “*03*”

  Width can be omitted or can be a number. If a number is specified are added to pad the length to match the number specified. If the number includes a leading 0, then zeroes are used instead of spaces for padding.

  .Precision can be omitted or can be either:

  - A number. For non-integral numeric types, the decimal portion of the output is expressed in at least this number of digits. For the string type, the output is truncated to this number of characters.
  - If precision is zero, nothing is printed for the corresponding argument
- Alphanumeric objects display stale values with a crosshatch shading effect.

Methods

“blur method” on page 413

“CancelChange method” on page 413

“CancelEvent method” on page 414

"click method" on page 417

"focus method" on page 427

"GetStyleProperty method" on page 445

Events

"OnActivate event" on page 497

"OnBlur event" on page 501

"OnChange event" on page 503

"OnClick event" on page 504

"OnContextMenu event" on page 505

"OnDataChange event" on page 507

"OnDblClick event" on page 508

"OnDeactivate event" on page 508

"OnFocus event" on page 512

"OnKeyDown event" on page 513

"OnKeyPress event" on page 513

"OnKeyUp event" on page 514

"OnMouseDown event" on page 516

"OnMouseEnter event" on page 517

"OnMouseLeave event" on page 517

"OnMouseMove event" on page 518

"OnMouseOut event" on page 519

"OnMouseOver event" on page 519

"OnMouseUp event" on page 520

"OnUpdate event" on page 538

Examples

The following example sets the display format to always show with a precision of two digits and have 3 leading digits padded with zeros:

```
alpha001.numericDisplayFormat="%07.2f"
```

The following example sets the display format to show a signed integer of 5 digits:

```
aplpha001.numericDisplayFormat="%+5d"
```

The following example demonstrates the use of the *currentValue* and *value* properties through the OnChange event of the alphanumeric object display script:

```
Sub alpha001_onchange
  dim Msg
  If alpha001.value < alpha001.currentValue Then
    Msg = "Value has been decreased"
    'insert script to handle value decrease
    ' …
  Else
    Msg = "Value has been increased"
    'insert script to handle value increase
    ' …
  End If
```

```
   window.external.MessageZoneText = Msg
 End Sub
```

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*.

Properties

Property	Description
AlarmZonePointID	Returns the ID of the current alarm point. The associated alarm appears in the Alarm Line. **Attention** 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. **Attention** 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. **Attention** 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.
AreSafeBrowseRightMouseClicksAllowed	Returns *True* if right-clicking a SafeBrowse object is allowed (that is, displays the shortcut menu). If not, it returns *False*. This property is not available in eServer premium.
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. **Attention** For compatibility with earlier releases, this property is available from this object. For future developments, you should access this property from the Station object. This property is not available in eServer premium.

Property	Description
`ConnectionFile`	Returns the filename (and path) of the Station setup file (`*.stn`) used for the current connection to the server. **Attention** For compatibility with earlier releases, this property is available from this object. For future developments, you should access this property from the Station object. This property is not available in eServer premium.
`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.
`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. This property is not available in eServer premium.
`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. **Attention** 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
MessageZoneText	Returns or sets the text that appears in the Station's Message Zone. ❗ **Attention** 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. This property is not available in eServer premium.
OnUpdateTime	Returns the duration, in milliseconds, for Station to execute any OnUpdate scripts when loading the current Station display.
OperatorID	Returns the ID of the operator. Returns the: • Operator ID if the Station is configured for operator-based security • Station number if the Station is configured for Station-based security ❗ **Attention** 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.
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. This property is not available in eServer premium.
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. This property is not available in eServer premium.
Parent	Returns the Station object. This property is not available in eServer premium.
parentElement	Returns the name of the element that launched a popup or faceplate within a multi-window Station display. ❗ **Attention** This property is not available within a single window station. This property is not available in eServer premium.
SecurityLevel	Returns the user's security level. The values are: • *0* = View Only • *1* = Ack Only • *2* = Operator • *3* = Supervisor • *4* = Engineer • *5* = Manager ❗ **Attention** 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 • *Non-zero* = Operator-based security **Attention** 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. **Attention** 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. **Attention** For compatibility with earlier releases, this property is available from this object. For future developments, you should access this property from the RuntimeStatus object. This property is not available in eServer premium.
StationDataRepository	Returns the StationDataRepository object. This property is not available in eServer premium.
StationNumber	Returns the Station's server connection number. **Attention** 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.
TemporaryMessageZoneText	Sets the text of a temporary message that appears in the Station's Message Zone for 5 seconds (after which it is cleared). **Attention** • 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. • 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 **Attention** 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).

Methods

"CancelOperatorCommand method" on page 415

"CancelResponse method" on page 415

"Connect method" on page 418

"CrossScreenInvocationPending method " on page 422

"DisplayExists method" on page 424

"ExecuteOperatorCommand method" on page 425

"FindFile method" on page 425

"FlagUserInteraction method" on page 427

"GetConnectionProperty method" on page 429

"GetResponse method" on page 442

"GetResponse2 method" on page 442

"GetResponse3 method" on page 443

"GetResponse4 method" on page 443

"InvokeCommand method" on page 451

"InvokeMenu2 method" on page 452

"InvokePopup method" on page 452

"KillTimer method" on page 456

"LogMessage method" on page 456

"MakeColor method" on page 457

"Quit method" on page 460

"Refresh method" on page 462

"RequestServerLRN method" on page 466

"RequestTask method" on page 467

"SendKeyPress method" on page 470

"Shell method" on page 492

"UserObjectNotify method" on page 495

Events

"OnActivated event" on page 498

"OnAlarm event" on page 498

"OnAppStartup event" on page 498

"OnAppStatusUpdate event" on page 499

"OnBeforeConnect event" on page 499

"OnBeforeOKBKeyPressed event" on page 500

"OnBlink event" on page 501

"OnConnect event" on page 504

"OnDictionaryValueChanged event" on page 509

"OnDisconnect event" on page 510

"OnDisplayNotFound event" on page 510

"OnMenu event" on page 515

"OnMenu2 event" on page 515

"OnOKBKeyPressed event" on page 520

"OnOperatorChange event" on page 521

"OnOperatorCommand event" on page 522

"OnOperatorCommand2 event" on page 522

"OnOperatorKey event" on page 523

"OnPageLoad event" on page 528

"OnPeriodicUpdate event" on page 528

"OnQuit event" on page 530

"OnResponse event" on page 534

"OnSilenceIKB event" on page 537

"OnTimer event" on page 537

"OnUserObjectNotify event" on page 539

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" on page 554

"Responding to Station events from another application" on page 570

Related topics

"Displaying messages to operators" on page 298
"Referencing objects" on page 301
"Event bubbling" on page 287
"Working with popups and faceplates" on page 305
"Using the Script Editor in Station" on page 338
"Writing Station-level scripts" on page 330

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*.

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 the Windows screen setting. For example, if the setting is 1280×1024, 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 1280×1024 resolution, the width can be up to 1280 pixels.
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” on page 424

“LPToDPX and LPToDPY methods” on page 457

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” on page 458

“SetCommandUIState method” on page 471

“SetTitleBarFormat method” on page 488

Events

None.

Remarks

- This object is not available in eServer premium.

Related topics

“Accessing Station-related properties in page-level scripts” on page 306

BasicTrend object

Description

The BasicTrend object represents a basic trend. (The Trend object represents a trend.)

The BasicTrend object contains a number of child objects that provide control over specific parts of the basic trend.

Properties

Property	Description
ChartArea	Returns the ChartArea object.
Legacy	Only used for backward compatibility. Returns the Legacy object, which replaces the pre-R300 HMIWeb trend.
Visible	Returns or sets the visibility of the basic trend, which can be: • *0* = Invisible • *1* = Visible

Methods

“GetPeriod method” on page 436

“GetSampleInterval method” on page 445

“SetPeriod method” on page 479

“SetSampleInterval method” on page 485

“SetSampleIntervaltoFastHistory method” on page 486

“SetTimeRange method” on page 487

Events

“OnPause event” on page 528

“OnResume event” on page 535

“OnScrollXAxis event” on page 535

“OnScrollYAxis event” on page 535

“OnTimeSelectorPositionChanged event” on page 537

“OnTrendLoadComplete event” on page 538

ChartArea object

Description

The ChartArea object is a child of the Trend object or BasicTrend 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.

Properties

Property	Description
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. Not applicable to a Basictrend 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

“CopyDataToClipboard method” on page 419

“GetXAxisToolbarItemVisible method” on page 447

“SetXAxisToolbarItemVisible method” on page 489

“GetDeadband method” on page 432

“SetDeadband method” on page 474

Events

None.

ChartToolbar object

Description

The ChartToolbar object is a child object of the Trend object.

Properties

Property	Description
font	Returns or sets the object's font-related properties. For example: `style.font = "italic small-caps bold 12pt serif"`
Visible	Sets the visibility of the chart toolbar.

Methods

"GetItemVisible method" on page 434

"SetItemVisible method" on page 477

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.
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.
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 check box's label.
lineColor	Returns or sets the object's line color.
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.
styleClass	Returns or sets the object's style.
textColor	Returns or sets the object's text color. The color can be either the RGB value (24-bit) or one of the predefined colors.
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" on page 413

"CancelChange method" on page 413

"CancelEvent method" on page 414

"click method" on page 417

"focus method" on page 427

"GetStyleProperty method" on page 445

Events

"OnActivate event" on page 497

"OnBlur event" on page 501

"OnChange event" on page 503

"OnClick event" on page 504

"OnContextMenu event" on page 505

"OnDataChange event" on page 507

"OnDblClick event" on page 508

"OnDeactivate event" on page 508

"OnFocus event" on page 512

"OnKeyDown event" on page 513

"OnKeyPress event" on page 513

"OnKeyUp event" on page 514

"OnMouseDown event" on page 516

"OnMouseEnter event" on page 517

"OnMouseLeave event" on page 517

"OnMouseMove event" on page 518

"OnMouseOut event" on page 519

"OnMouseOver event" on page 519

"OnMouseUp event" on page 520

"OnUpdate event" on page 538

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.
disabled	Returns or sets operator control of the object, the value is either *True* or *False*.

Property	Description
ElementINFState	Returns whether a value is infinity or not. Returns *0* for non-infinity values, *-1* for negative infinity values or *1* for positive infinity values.
fillColor	Returns or sets the object's fill color.
fillColorBlink	Returns or sets the blinking state of the object's fill color. The value is either *True* or *False*.
GetOptionList	Returns the list of values within the respective combo box. The combo box list is populated when the combo box is clicked. In order to access the combo box list you call the *combobox.click* method to populate the list. You must allow for a time delay in communicating with the server in order for the server to populate the combo box list.
id	Returns the object's name.
IsElementNaN	Returns *True* if the value of the element is not a number (NaN), otherwise, returns *False*.
label	Returns or sets the check box's label.
lineColor	Returns or sets the object's line color.
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.
styleClass	Returns or sets the object's style.
textColor	Returns or sets the object's text color. The color can be either the RGB value (24-bit) or one of the predefined colors.
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 (a string) displayed in the combo box.

Remarks

- Combobox objects display stale values with a crosshatch shading effect.

Methods

“AddString method” on page 412

“DeleteString method” on page 423

“ResetContent method” on page 469

“CancelChange method” on page 413

“CancelEvent method” on page 414

“click method” on page 417

“GetStyleProperty method” on page 445

Events

“OnActivate event” on page 497

“OnBlur event” on page 501

“OnChange event” on page 503

“OnClick event” on page 504

“OnContextMenu event” on page 505

“OnDataChange event” on page 507

“OnDblClick event” on page 508

“OnDeactivate event” on page 508

“OnFocus event” on page 512

“OnKeyDown event” on page 513

“OnKeyPress event” on page 513

“OnKeyUp event” on page 514

“OnMouseDown event” on page 516

“OnMouseEnter event” on page 517

“OnMouseLeave event” on page 517

“OnMouseMove event” on page 518

“OnMouseOut event” on page 519

“OnMouseOver event” on page 519

“OnMouseUp event” on page 520

“OnUpdate event” on page 538

Example

The following example accesses the combobox list values by declaring 2 functions (in the General Section). myFunction calls myFunction2() using a timer and manipulation of the combobox list occurs in myFunction2().

```
Dim intTimerID
Function myFunction()
    .
    .
    .
    combobox001.click
    Dim
    varList
    intTimerID = window.setInterval ("myFunction2 ()",200)
    .
    .
    .
End Function
Function myFunction2()
    dim varList
    combobox001.GetOptionList varList
    combobox001.click
    .
    .
    .
    window.clearInterval (intTimerID)
End Function
```

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.
- If you have a multi-window Station, the call-up performance of a display might be degraded if it makes extensive use of the Dictionary.

Methods

"Add method" on page 409

"Item method" on page 455

"Remove method" on page 463

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
autoReloadContent	Returns or sets automatic reloading of custom properties when there is a change in the value of one or more properties. The values are: • *True* = Automatic reload is enabled (default) • *False* = Automatic reloading is disabled For performance reasons, if you want to change the value of several custom properties at the same time, you should set this to False before changing the values, and then set it back to True after changing them.
Count	Returns the number of properties in the repository.
Item(index)	Returns the custom property with the specified *index*. The index is an integer, and index of the first property in the collection is *0*.

Methods

"Add method" on page 409

"GetValue method" on page 447

"Item method" on page 455

"PutValue method" on page 460

"Remove method" on page 463

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/en-us/library/ms535863(VS.85).aspx.

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*. **Note** that this does not cancel the event action.
keyCode	Returns or sets the Unicode 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 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. **Note** that this does not cancel the event bubbling.
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

- In script, you must access the Event object through the window object.
- 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.

Examples

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 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.returnValue = False
```

Related topics

"Referencing objects" on page 301

"Event bubbling" on page 287

EventBar object

Description

The EventBar object is a child object of the Trend object (but not the BasicTrend object). It contains properties to control and manipulate the event bar in the chart area when the trend has events visible.

Properties

Property	Description
BackgroundColor	Returns or sets the background color of the event bar.

Property	Description
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" on page 430

"GetDataTipVisible method" on page 432

"HideAllDataTips method" on page 450

"HideDataTips method" on page 450

"SetDataTipArrowColors method" on page 473

"SetDataTipVisible method" on page 474

Events

None.

Group object

Properties

Property	Description
all *children*	These are standard DHTML collections that return a reference to the collection of child objects.
*Data** *Quality** *AlarmSeverity*	These properties only exist if one or more point parameters are listed on the object's Script Data tab.
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.

Property	Description
styleClass	Returns or sets the object's style.
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” on page 445

Events

“OnActivate event” on page 497

“OnBlur event” on page 501

“OnClick event” on page 504

“OnContextMenu event” on page 505

“OnDataChange event” on page 507

“OnDblClick event” on page 508

“OnDeactivate event” on page 508

“OnFocus event” on page 512

“OnKeyDown event” on page 513

“OnKeyPress event” on page 513

“OnKeyUp event” on page 514

“OnMouseDown event” on page 516

“OnMouseEnter event” on page 517

“OnMouseLeave event” on page 517

“OnMouseMove event” on page 518

“OnMouseOut event” on page 519

“OnMouseOver event” on page 519

“OnMouseUp event” on page 520

“OnUpdate event” on page 538

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.
disabled	Returns or sets operator control of the object, the value is either *True* or *False*.
HREF	Returns or sets the URL.
id	Returns the object's name.
innerText	Returns or sets the object's text.

Property	Description
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.
styleClass	Returns or sets the object's style.
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” on page 413

“click method” on page 417

“focus method” on page 427

“GetStyleProperty method” on page 445

Events

“OnActivate event” on page 497

“OnBlur event” on page 501

“OnClick event” on page 504

“OnContextMenu event” on page 505

“OnDataChange event” on page 507

“OnDblClick event” on page 508

“OnDeactivate event” on page 508

“OnFocus event” on page 512

“OnKeyDown event” on page 513

“OnKeyPress event” on page 513

“OnKeyUp event” on page 514

“OnMouseDown event” on page 516

“OnMouseEnter event” on page 517

“OnMouseLeave event” on page 517

“OnMouseMove event” on page 518

“OnMouseOut event” on page 519

“OnMouseOver event” on page 519

“OnMouseUp event” on page 520

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.
disabled	Returns or sets operator control of the object, the value is either *True* or *False*.
ElementINFState	Returns whether a value is infinity or not. Returns *0* for non-infinity values, *-1* for negative infinity values or *1* for positive infinity values.
fillColor	Returns or sets the object's fill color.
fillColorBlink	Returns or sets the blinking state of the object's fill color. The value is either *True* or *False*.
fillDirection	Returns or sets the object's fill direction. Value is either: • For horizontal indicators, *Left* or *Right*. • For vertical indicators, *Up* or *Down*.
fillStyle	Returns or sets how the object is animated. Values are either *Bar*, *Pointer*, *Hollow Pointer*, or *Line*.
id	Returns the object's name.
IsElementNaN	Returns *True* if the value of the element is not a number (NaN), otherwise, returns *False*. If *True*, the indicator object displays a gray cross.
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 colors.
lineColor	Returns or sets the object's line color.
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.
positiveDirection	Returns or sets the fill direction of the object for a positive value. Value is either *Up*, *Down*, *Left* or *Right*.
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.
styleClass	Returns or sets the object's style.
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).
visibleRangeHi *visibleRangeLo*	Return or set 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*.

Remarks

- Indicator objects display stale values with a crosshatch shading effect.

Methods

"blur method" on page 413

"focus method" on page 427

"GetStyleProperty method" on page 445

Events

"OnActivate event" on page 497

"OnBlur event" on page 501

"OnClick event" on page 504

"OnContextMenu event" on page 505

"OnDataChange event" on page 507

"OnDblClick event" on page 508

"OnDeactivate event" on page 508

"OnFocus event" on page 512

"OnKeyDown event" on page 513

"OnKeyPress event" on page 513

"OnKeyUp event" on page 514

"OnMouseDown event" on page 516

"OnMouseEnter event" on page 517

"OnMouseLeave event" on page 517

"OnMouseMove event" on page 518

"OnMouseOut event" on page 519

"OnMouseOver event" on page 519

"OnMouseUp event" on page 520

"OnUpdate event" on page 538

Example

rangeHi and *rangeLo* indicate the actual maximum and minimum values shown on the indicator (corresponding to the top and the bottom of the indicator respectively).*visibleRangeHi* and *visibleRangeLo* are percentages of the actual range (*rangeHi* and rangeLo) which the indicator displays. For example:

An indicator is set to have a *rangeHi* of 100 and a *rangeLo* of 20. If the *visibleRangeHi* and *visibleRangeLo* are left at their default values of 100 and 0 then the indicator will show all values between 100 and 20 (if the actual value were greater than a 100 the indicator would draw as full and if it were below 20 it would draw as empty.

If the *visibleRangeHi* and *visibleRangeLo* are set to 80 and 20 respectively the indicator will now display values between 84 and 36 ((100-20)*.8+20 and (100-20)*.2+20).

Note that the ranges set via the property pages for the indicator refer to the *visibleRangeHi* and *visibleRangeLo* rather than *rangeHi* and *rangeLo*.

Legacy object

Description

The Legacy object is only provided to support backward compatibility, and replaces the pre-R300 HMIWeb trend. It is a child object of the Trend object and BasicTrend object

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.
scrollBarsVisible	Returns or sets the visibility of the scroll bars, 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.
- If you want to control the visibility of the object, you do it through the style object, for example:

```
trend001.style.visibility = "hidden"
```

Methods

"AddNewPlot method" on page 410

"AddPlotData method" on page 411

"ClearPlotData method" on page 416

"Copy method" on page 419

"GetPlotColor method" on page 437

"GetPlotDiscrete method" on page 437

"GetPlotMarkerStyle method" on page 438

"GetPlotVisible method" on page 440

"GetReferenceValues method" on page 441

"GetStyleProperty method" on page 445

"Redraw method" on page 462

"RemovePlot method" on page 464

"RemovePlotData method" on page 465

"SetPlotColor method" on page 480

"SetPlotDiscrete method" on page 480

"SetPlotMarkerStyle method" on page 481

"SetPlotPen method" on page 482

"SetPlotRange method" on page 483

"SetPlotStepMode method" on page 484

"SetPlotVisible method" on page 485

"UnZoom method" on page 495

Events

"Legacy_OnChartLoadComplete event" on page 541

"Legacy_OnChartUnZoomed event" on page 541

"Legacy_OnChartScroll event" on page 542

"Legacy_OnChartZoom event" on page 542

"Legacy_OnReferenceLineSet event" on page 544

Legend object

Description

The Legend object is a child object of the Trend object. It contains properties to control and manipulate the legend associated with the trend's chart.

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.

Property	Description
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.
RowsVisible	Returns or sets the number of rows that are simultaneously visible. (Users can use the scroll bar to view other rows.) Note that increasing/decreasing the number of rows that are visible will decrease/increase the size of the plot area.
Unit	Returns the unit of the plot in the given row.
Visible	Returns or sets the visibility of the legend.

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.

Methods

"SetNodePoint method" on page 478

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.

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.

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.
commandElement	Only applicable to a faceplate. Sets the faceplate's command element, in response to a OnRequestCommandElement event. For details about when to use this property, see 'Faceplate auto-selection' in the *Server and Client Configuration Guide*.
hasFocus	Returns *True* if the page has windows focus; otherwise returns *False*.
modeElement	Only applicable to a faceplate. Sets the faceplate's mode element, in response to a OnRequestModeElement event. For details about when to use this property, see 'Faceplate auto-selection' in the *Server and Client Configuration Guide*.
outputElement	Only applicable to a faceplate. Sets the faceplate's output element, in response to a OnRequestOutPutElement event. For details about when to use this property, see 'Faceplate auto-selection' in the *Server and Client Configuration Guide*.
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.
setpointElement	Only applicable to a faceplate. Sets the faceplate's set point element, in response to a OnRequestSetPointElement event. For details about when to use this property, see 'Faceplate auto-selection' in the *Server and Client Configuration Guide*.
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.

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” on page 445

“GetStyleClassProperty method” on page 446

Events

"OnActivate event" on page 497

"OnActivated event" on page 498

"OnAlarm event" on page 498

"OnAppStatusUpdate event" on page 499

"OnBeforeConnect event" on page 499

"OnBlink event" on page 501

"OnBlur event" on page 501

"OnClick event" on page 504

"OnContextMenu event" on page 505

"OnDblClick event" on page 508

"OnDeactivate event" on page 508

"OnDictionaryValueChanged event" on page 509

"OnFocus event" on page 512

"OnKeyDown event" on page 513

"OnKeyPress event" on page 513

"OnKeyUp event" on page 514

"OnMenu2 event" on page 515

"OnMouseDown event" on page 516

"OnMouseEnter event" on page 517

"OnMouseLeave event" on page 517

"OnMouseMove event" on page 518

"OnMouseOut event" on page 519

"OnMouseOver event" on page 519

"OnMouseUp event" on page 520

"OnOperatorChange event" on page 521

"OnOperatorCommand2 event" on page 522

"OnOperatorKey event" on page 523

"OnPageComplete event" on page 527

"OnPeriodicUpdate event" on page 528

"OnRequestCommandElement event" on page 531 (only applicable to a faceplate)

"OnRequestModeElement event" on page 533 (only applicable to a faceplate)

"OnRequestSetPointElement event" on page 533 (only applicable to a faceplate)

"OnRequestOutPutElement event" on page 534 (only applicable to a faceplate)

"OnResponse event" on page 534

"OnUnload event" on page 538

"OnUserObjectNotify event" on page 539

"OnViewportChanged event" on page 540

Tip

The *OnViewportChanged* event is only applicable on Pan and Zoom displays.

Related topics

"Referencing objects" on page 301
"Event bubbling" on page 287
"Using HMIWeb error handlers" on page 333

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.
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.
styleClass	Returns or sets the object's style.
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" on page 445

Events

"OnActivate event" on page 497

"OnBlur event" on page 501

"OnClick event" on page 504

"OnContextMenu event" on page 505

"OnDataChange event" on page 507

"OnDblClick event" on page 508

"OnDeactivate event" on page 508

"OnFocus event" on page 512

"OnKeyDown event" on page 513

"OnKeyPress event" on page 513

"OnKeyUp event" on page 514

"OnMouseDown event" on page 516

"OnMouseEnter event" on page 517

"OnMouseLeave event" on page 517

"OnMouseMove event" on page 518

"OnMouseOut event" on page 519

"OnMouseOver event" on page 519

"OnMouseUp event" on page 520

Plot object

Description

The Plot object represents a specific plot in a trend or basic trend. Because a trend/basic trend can have up to 32 plots, it can have up to 32 Plot objects.

The Plot object is a child of the Plots object.

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" on page 412

"GetLabels method" on page 435

"GetPen method" on page 435

"GetYAxisScale method" on page 448

"GetYAxisUnzoomedScale method" on page 449

"SetLabels method" on page 477

"SetPen method" on page 479

"SetYAxisScale method" on page 490

"SetYAxisUnzoomedScale method" on page 491

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.

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

“Add method” on page 409

“Autoscale method” on page 412

“GetBoundPlot method” on page 428

“GetXAxisScale method” on page 448

“GetXAxisUnzoomedScale method” on page 449

“GetYAxisScale method” on page 448

“GetYAxisUnzoomedScale method” on page 449

“Remove method” on page 463

“SetXAxisScale method” on page 489

“SetBoundPlot method” on page 470

“SetXAxisUnzoomedScale method” on page 490

“SetYAxisScale method” on page 490

“SetYAxisUnzoomedScale method” on page 491

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.

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 method" on page 423

"GetGridLinePen method" on page 433

"GetZoomRegion method" on page 450

"IncreaseZoomLevel method" on page 451

"SetGridLinePen method" on page 475

"SetZoomRegion method" on page 491

"UnZoom method" on page 495

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.

Tip

Be aware that the PlotDataTip object only represents the ToolTip for the plot data section of the chart area in a trend or basic trend graph.

The ToolTip for the reference line is set in script using the ReferenceCursor object.

The ToolTip for the event bar is set in script using the EventBar object.

Properties

Property	Description
DateFormat	The coded string format for any date/time information in the plot data tips on the chart. By default, if not set (remaining empty), the regional (system locale) settings for the computer are used. The data tip date format is a string that contains specifiers that are replaced at runtime with an equivalent value. Specifier codes are preceded by a percent sign (%). Characters that do not begin with % are used unchanged. The specifier codes for the DateFormat property are: • %a = Abbreviated weekday name • %A = Full weekday name • %b = Abbreviated month name • %B = Full month name • %c = Date and time representation appropriate for locale • %d = Day of month as decimal number (01 – 31) • %H = Hour in 24-hour format (00 – 23) • %I = Hour in 12-hour format (01 – 12) • %j = Day of year as decimal number (001 – 366) • %m = Month as decimal number (01 – 12) • %M = Minute as decimal number (00 – 59) • %p = Current locale's A.M./P.M. indicator for 12-hour clock • %S = Second as decimal number (00 – 59) • %U = Week of year as decimal number, with Sunday as first day of week (00 – 53) • %w = Weekday as decimal number (0 – 6; Sunday is 0) • %W = Week of year as decimal number, with Monday as first day of week (00 – 53) • %x = Date representation for current locale • %X = Time representation for current locale • %y = Year without century, as decimal number (00 – 99) • %Y = Year with century, as decimal number • %z, %Z = Either the time-zone name or time zone abbreviation, depending on registry settings; no characters if time zone is unknown • %% = Percent sign
Enable	True (default) or False. The data tips visibility state. If set to *False*, the data tips for both plot data ToolTips and the reference line data ToolTip are disabled and not visible.
Format	The coded string format of the plot data tips which specifies what information is displayed and the order of that information. By default, if not set (remaining empty), the coded string "“%p : %y %Y” is used. The data tip format is a string that contains specifiers that are replaced at runtime with an equivalent value. Specifier codes are preceded by a percent sign (%). Characters that do not begin with % are used unchanged. The specifier codes for the Format property are: • %p = Plot name • %y = Y-axis value • %Y = Y-axis units • %x = X-axis value • %X = X-axis units • %% = Percent sign

Property	Description
UseCustomDataTips	True or False (default). Determines the source of data tips for both plot data ToolTips and the reference line data ToolTip. By default, when set to *False*, uses the settings of the Format and DateFormat properties of the PlotDataTip object. If set to *True*, you should script custom dynamic data tips by handling data tip events and specifying what is to be displayed. **Attention** Be aware that setting the UseCustomDataTips property to *True* will cause the ToolTips for both the plots and the reference line to ignore the default and custom formatting properties, whether set or not, and rely solely upon formatting provided respectively by scripting of the OnRequestDataTip event and the OnRequestReferenceDataTip event of the Trend object. If no suitable event handling is provided, no plot data ToolTips or reference data ToolTip will be shown.

Methods

None.

Events

None.

Example

You can change the format of the plot data tip by setting the Format property on the PlotDataTip object. For example, if you have a plot configured whose pointname is 'BoilerTemp' and you formatted the data ToolTip using script as follows:

```
trend001.ChartArea.PlotDataTip.Format = "BoilerInfo: %p %y %Y"
```

- results in the following data ToolTip 'BoilerInfo: BoilerTemp 35 degrees.'

You can set the date format of the plot data tip by setting the DateFormat property on the PlotDataTip object. For example, if you formatted the DateFormat in script as follows:

```
trend001.ChartArea.PlotDataTip.DateFormat = "%d/%m/%y"
```

- results in a formatted ToolTip showing the date in day, month, year numeric pair format (for example, '30/12/2012').

Popup object

Properties

None.

Remarks

- You use *HDXPopupBehavior* to access popup objects. For example, this calls up a popup called 'Status.htm' (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
  ```

- To display a custom shortcut menu, you would use the *InvokeShortcutMenu* method of the *HDXPopupBehavior* popup object. For example:

  ```
  HDXPopupBehavior.InvokeShortcutMenu "myShortcutMenu.dita", window.event.x, window.event.y
  ```

Methods

"CreatePopupWindow method" on page 420

"CreatePopupWindow2 method" on page 421

"FindPopup method" on page 426

"InvokeShortcutMenu method" on page 454

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.
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.
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.
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.
styleClass	Returns or sets the object's style.
textColor	Returns or sets the object's text color. The color can be either the RGB value (24-bit) or one of the predefined colors.
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" on page 413

"CancelChange method" on page 413

"CancelEvent method" on page 414

"click method" on page 417

"focus method" on page 427

"GetStyleProperty method" on page 445

Events

"OnActivate event" on page 497

"OnBlur event" on page 501

"OnClick event" on page 504

"OnContextMenu event" on page 505

"OnDataChange event" on page 507

"OnDblClick event" on page 508

"OnDeactivate event" on page 508

"OnFocus event" on page 512

"OnKeyDown event" on page 513

"OnKeyPress event" on page 513

"OnKeyUp event" on page 514

"OnMouseDown event" on page 516

"OnMouseEnter event" on page 517

"OnMouseLeave event" on page 517

"OnMouseMove event" on page 518

"OnMouseOut event" on page 519

"OnMouseUp event" on page 520

"OnMouseOver event" on page 519

"OnMouseUp event" on page 520

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.

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	True (default) or False. Determines the placement of the reference cursor. If set to *False*, 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. When set to *True*, 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" on page 431

"GetLineStyle method" on page 435

"GetReferenceValues method" on page 441

"SetDataTipFormat method" on page 473

"SetLineStyle method" on page 478

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*.

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.
ConsoleName	Returns the name of the Console Station.
Date	Returns the date that appears in the Status Bar.

FieldColor(fieldid)
FieldState(fieldid)
FieldText(fieldid)
	Return the color, state, and text of the specified box in the Status Bar.

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 • *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.
ServerVersion	Returns the version number of the server to which the Station is connected.
SOMUserStatus	Returns the status of Signon Manager: • 0 = Signon Manager is not in use. • 1 = Signon Manager is not signed in. (None mode). • 2 = Signon Manager is signed in. (Normal mode). • 3 = Signon Manager is signed in and in Override mode.
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 Bar.

Methods

None.

Events

None.

Remarks

- This object is not available in eServer premium.

Status Bar 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.

Box (identifier)	Status values
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.
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.

Related topics

"Accessing Station-related properties in page-level scripts" on page 306

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.

Property	Description
customProperties.Item(n).name	Returns the name of the n^{th} custom property.
customProperties.Item(n).value	Returns the value of the n^{th} custom property.
*Data** *Quality** *AlarmSeverity*	These properties only exist if one or more point parameters are listed on the object's **Script Data** tab.
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 shapefile object itself has scripts attached. If not, returns False. As this is a property of the shapefile object, does not recognize display or general scope scripts.
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*.
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.
styleClass	Returns or sets the object's style.
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" on page 430

"GetStyleProperty method" on page 445

"Objects method" on page 459

"ReloadContent method" on page 463

"SetCustomProperty method" on page 472

Events

"OnActivate event" on page 497

"OnBlur event" on page 501

"OnClick event" on page 504

"OnContextMenu event" on page 505

"OnDataChange event" on page 507

"OnDblClick event" on page 508

"OnDeactivate event" on page 508

"OnFocus event" on page 512

"OnKeyDown event" on page 513

"OnKeyPress event" on page 513

"OnKeyUp event" on page 514

"OnMouseDown event" on page 516

"OnMouseEnter event" on page 517

"OnMouseLeave event" on page 517

"OnMouseMove event" on page 518

"OnMouseOut event" on page 519

"OnMouseOver event" on page 519

"OnMouseUp event" on page 520

"OnOperatorKey event" on page 523 (only available from within the shape file)

"OnShapeLoad event" on page 536 (only available from within the shape file)

"OnShapeUnload event" on page 536 (only available from within the shape file)

"OnUpdate event" on page 538 (only applicable to animated shape sequences)

Related topics

"Working with shapes" on page 303
"Objects method" on page 459

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`.

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).

Property	Description
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.
TemporaryMessageZoneText	Sets the text of a temporary message that appears in the Station's Message Zone for 5 seconds (after which it is cleared). **Attention** 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” on page 415

“Connect method” on page 418

“CreateTimer method” on page 421

“CrossScreenInvocationPending method ” on page 422

“DisplayExists method” on page 424

“ExecuteOperatorCommand method” on page 425

“FindFile method” on page 425

“FlagUserInteraction method” on page 427

“GetConnectionProperty method” on page 429

“GetResponse method” on page 442

“InvokeCommand method” on page 451

“InvokePopup method” on page 452

“KillTimer method” on page 456

“LogMessage method” on page 456

“MessageBox method” on page 458

“Quit method” on page 460

“RequestServerLRN method” on page 466

“RequestTask method” on page 467

“SetCommandUIState method” on page 471

“SetTitleBarFormat method” on page 488

“SetUserVersionInfo method” on page 488

“Shell method” on page 492

Events

“OnAlarm event” on page 498

“OnAppStartup event” on page 498

“OnAppStatusUpdate event” on page 499

“OnBeforeConnect event” on page 499

“OnBeforeOKBKeyPressed event” on page 500

“OnBlink event” on page 501

“OnConnect event” on page 504

“OnCustomEvent event” on page 505

“OnDictionaryValueChanged event” on page 509

“OnDisconnect event” on page 510

“OnDisplayNotFound event” on page 510

“OnMenu event” on page 515

“OnOKBKeyPressed event” on page 520

“OnOperatorChange event” on page 521

“OnOperatorCommand event” on page 522

“OnPageLoad event” on page 528

“OnPeriodicUpdate event” on page 528

“OnQuit event” on page 530

“OnResponse event” on page 534

“OnSilenceIKB event” on page 537

“OnTimer event” on page 537

Related topics

“Displaying messages to operators” on page 298
“Working with popups and faceplates” on page 305
“Accessing Station-related properties in page-level scripts” on page 306
“MessageBox method” on page 458
“showCallout method” on page 493

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
autoReloadContent	Returns or sets automatic reloading of custom properties when there is a change in the value of one or more properties. The values are: • *True* = Automatic reload is enabled (default) • *False* = Automatic reloading is disabled For performance reasons, if you want to change the value of several custom properties at the same time, you should set this to False before changing the values, and then set it back to True after changing them.
Count	Returns the number of properties in the repository.

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.

Methods

"Add method" on page 409

"GetValue method" on page 447

"PutValue method" on page 460

"Remove method" on page 463

Events

None.

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.

Property	Description
Count	Returns the number of windows in the StationWindows collection. This corresponds to the number of windows currently displayed.

Methods

"Item method" on page 455

Events

None.

Examples

To access the StationWindows collection from script on a display page, you must use the following structure:

```
window.external.Parent.StationWindows
```

For example, to find the name of the display in the current active window you use the following:

```
window.external.Parent.StationWindows.ActiveWindow.CurrentPage
```

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
  If(stationwindow.document.Title = "mypage.htm")
    Then
    MsgBox
  stationwindow.document.getElementById("myobject").Value
  End If
Next
```

Related topics

"Scripting in a multi-window environment" on page 324

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://msdn2.microsoft.com/library/ms535870.

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.
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" on page 560

Related topics

"Writing Station-level scripts" on page 330
"Using event bubbling to reduce script effort" on page 577

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.
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.
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.
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.
styleClass	Returns or sets the object's style.
textColor	Returns or sets the object's text color. The color can be either the RGB value (24-bit) or one of the predefined colors.
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 text displayed in the object.

Methods

"blur method" on page 413

"click method" on page 417

"focus method" on page 427

"GetStyleProperty method" on page 445

Events

"OnActivate event" on page 497

"OnBlur event" on page 501

"OnChange event" on page 503

"OnClick event" on page 504

"OnContextMenu event" on page 505

"OnDataChange event" on page 507

"OnDblClick event" on page 508

"OnDeactivate event" on page 508

"OnFocus event" on page 512

"OnKeyDown event" on page 513

"OnKeyPress event" on page 513

"OnKeyUp event" on page 514

"OnMouseDown event" on page 516

"OnMouseEnter event" on page 517

"OnMouseLeave event" on page 517

"OnMouseMove event" on page 518

"OnMouseOut event" on page 519

"OnMouseOver event" on page 519

"OnMouseUp event" on page 520

"OnUpdate event" on page 538

"onpropertychange event" on page 529

TitleToolbar object

Description

The TitleToolbar object is a child object of the Trend object.

Properties

Property	Description
Visible	Sets the visibility of the chart toolbar.

Methods

"GetItemVisible method" on page 434

"SetItemVisible method" on page 477

Events

None.

Trend object

Description

The trend object represents a trend. (The BasicTrend object represents a basic trend.)

The trend object contains a number of child objects that provide control over specific parts of the trend.

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	Only used for backward compatibility. Returns the Legacy object, which replaces the pre-R300 HMIWeb trend.
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" on page 445

"RuntimeDataRequest method" on page 469

"SetPeriod method" on page 479

"SetSampleInterval method" on page 485

Events

"OnEventDataTipDisplayed event" on page 511

"OnPause event" on page 528

"OnPlotDetailsChanged event" on page 529

"OnPlotDataBufferFull event" on page 528

"OnReferenceCursorSet event" on page 530

"OnRequestDataTip event" on page 532

"OnRequestEventInfo event" on page 532

"OnResume event" on page 535

"OnScrollXAxis event" on page 535

"OnScrollYAxis event" on page 535

"OnTimeSelectorPositionChanged event" on page 537

"OnTrendLoadComplete event" on page 538

"OnUnZoomed event" on page 539

"OnZoom event" on page 541

Vector Graphic object

Description

A vector graphic object is a 'line-based' object such as a rectangles or curve.

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.

`fillColor`	Returns or sets the object's fill color.
`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 colors.

Property	Description
`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 colors. Note that the color is only visible if the LevelFilled property is `True`.
`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.
`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 Other properties typically set by the style object are specified using object-specific properties.
`styleClass`	Returns or sets the object's style.

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"*

Methods

“GetStyleProperty method” on page 445

Events

“OnActivate event” on page 497

“OnBlur event” on page 501

“OnChange event” on page 503

“OnClick event” on page 504

“OnContextMenu event” on page 505

“OnDataChange event” on page 507

“OnDblClick event” on page 508

“OnDeactivate event” on page 508

“OnFocus event” on page 512

“OnKeyDown event” on page 513

“OnKeyPress event” on page 513

“OnKeyUp event” on page 514

“OnMouseDown event” on page 516

“OnMouseEnter event” on page 517

“OnMouseLeave event” on page 517

“OnMouseMove event” on page 518

“OnMouseOut event” on page 519

“OnMouseOver event” on page 519

“OnMouseUp event” on page 520

“OnUpdate event” on page 538

window object

Description

Represents an open window.

Properties

Property	Description
external	Related to the Application object.
event	Related to the Event object.

Methods

“setTimeout method” on page 486

"setInterval method" on page 476

"clearInterval method" on page 416

"clearTimeout method" on page 417

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 chart area.

Properties

Property	Description
ApplyScrollLimits	Returns or sets the scrolling limits. The possible values are: • *0* = (default) The trend can scroll past the limit of available data.For example when scrolling forward this allows you to scroll into the future an amount equal to half of the visible range of the trend. If viewing a 1 hour period you will be able to scroll 30 minutes into the future. • *1* = The trend can scroll to the limits of the contained data.For example if the trend is currently viewing 12:00am to 1:00pm yet the trend stores data from 10:00am to 2:00pm, then you are able to scroll from 10:00am to 2:00pm. You cannot scroll past the limits of the contained data. This setting prevents further historical data being automatically retrieved when scrolling in the trend.
BackgroundColor	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.
ScrollBarVisible	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” on page 433

“GetRange method” on page 440

“Scroll method” on page 469

“SetInPlaceEditing method” on page 475

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 chart area.

Properties

Property	Description
ApplyScrollLimits	Returns or sets the scrolling limits. The possible values are: • *0* = Allowed to scroll half a page past the data • *1* = Allowed to scroll to the edges of the data • *2* = (default) Allowed to scroll only when zoomed
BackgroundColor	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

Property	Description
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.
MarkerLabelColor	Returns or sets the color of the axis marker text.
ScrollBarVisible	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” on page 433

“GetRange method” on page 440

“Scroll method” on page 469

“SetInPlaceEditing method” on page 475

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.

Related topics

Add method

Applicable to

"Dictionary object" on page 365

"DisplayDataRepository object" on page 366

"StationDataRepository object" on page 394

Description

Adds a new item to the object.

Syntax

For the Dictionary object:

```
Add name, value, persist
```

For the StationDataRepository and DisplayDataRepository objects:

```
Add name, type, value, description
```

Part	Description
name	The name of the item. The name is case-sensitive.
type	The type of item, either *point*, *parameter* or *value*.
value	The value of the item. The value can be of any form, such as text or numeric.
persist	A boolean value (*true* or *false*) to specify if the dictionary entry should be persistent on the computer running Station. The default value is *false* (the dictionary entry is non-persistent).
description	A description of the item.

Return values

None.

Example

This example adds an item 'Voltage' to the Dictionary and sets its value to '12.3.' As the persist boolean is not specified, the 'Voltage' entry is non-persistent.

```
window.external.Dictionary.Add "Voltage", 12.3
```

This example adds an item 'Voltage' to the Dictionary, sets its value to '12.3,' and specifies that the 'Voltage' be persistent on the computer running Station.

```
window.external.Dictionary.Add "Voltage", 12.3, true
```

This example adds a system custom property "Custom1" to the Station Data Repository.

```
window.external.stationDataRepository.Add "Custom1", "Point", "sinewave", " "
```

This example adds a system custom property 'Custom1' to the display Data Repository.

```
 displayDataRepository.Add "Custom1", "Point", "sinewave", " "
```

AddNewPlot method

Applicable to

"Legacy object" on page 374

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.
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” on page 374

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.

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” on page 363

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” on page 381

“Plots object” on page 382

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.

blur method

Applicable to

“Alphanumeric object” on page 350

“Checkbox object” on page 362

“Combobox object” on page 363

“Hyperlink object” on page 370

“Indicator object” on page 372

“Picture object” on page 380

“Pushbutton object” on page 386

“Shape object” on page 390

“Text object” on page 397

“Vector Graphic object” on page 400

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” on page 350

“Checkbox object” on page 362

“Combobox object” on page 363

“Pushbutton object” on page 386

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 alpha001_onchange
  'calculate current limit
  Dim LimitVal
  'insert script to assign current limit to LimitVal variable
  ' …
  'evaluate value against current limit
  If alpha001.value > LimitVal Then
    window.external.MessageZoneText = "Value must be below " _
      & CStr(Limit)
      alpha001.CancelChange
      window.event.returnValue = "false"
  End If
End Sub
```

Related topics

“Writing Station-level scripts” on page 330

CancelEvent method

Applicable to

“Alphanumeric object” on page 350

“Checkbox object” on page 362

“Combobox object” on page 363

“Pushbutton object” on page 386

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
```

Related topics

"Writing Station-level scripts" on page 330

CancelOperatorCommand method

Applicable to

"Application object" on page 353

"Station object" on page 392

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" on page 353

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.

clearInterval method

Applicable to

"window object" on page 402

Description

Cancels the interval previously started using the setInterval method.

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.

ClearPlotData method

Applicable to

"Legacy object" on page 374

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” on page 402

Description

Cancels a timeout that was set with the setTimeout method.

Syntax

```
window.clearTimeout(iTimeoutID)
```

Part	Description
iTimeoutID	A required integer that specifies the timeout setting returned by a previous call to the setTimeout method.

Return values

None.

click method

Applicable to

“Alphanumeric object” on page 350

“Checkbox object” on page 362

“Combobox object” on page 363

“Hyperlink object” on page 370

“Indicator object” on page 372

“Page object” on page 378

“Picture object” on page 380

“Pushbutton object” on page 386

“Shape object” on page 390

“Text object” on page 397

“Vector Graphic object” on page 400

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” on page 353

“Station object” on page 392

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:\ProgramData\Honeywell\Experion PKS\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” on page 374

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 scroll bars, 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” on page 361

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" on page 385

Description

Calls up a popup window or faceplate.

Syntax

```
CreatePopupWindow DisplayPath, Xpos, Ypos[, PositionType]
```

Part	Description
DisplayPath	Required. 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*	Required. 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 positioning rules as specified in the *PositionType* parameter.
PositionType	Optional. 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
```

CreatePopupWindow2 method

Applicable to

“Popup object” on page 385

Description

Calls up a popup window or faceplate.

Syntax

```
CreatePopupWindow2 DisplayPath, Xpos, Ypos[, PositionType], vbPersist
```

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
vbPersist	Specifies if the popup or faceplate is docked: • *True* = Docked • *False* = Undocked Note that vbPersist is not applicable in a multi-window Station.

Return values

None.

Example

This example calls up a popup called 'Status' (which is in one of Station's display folders), places it at the top-left of the display and docks the popup.

```
HDXPopupBehavior.CreatePopupWindow2 "Status.htm", 1, 1, 0, True
```

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.CreatePopupWindow2 "sysdtlana_fp.htm?currentpoint=tank1", 0, 0, 1, True
```

CreateTimer method

Applicable to

“Station object” on page 392

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
```

CrossScreenInvocationPending method

Applicable to

“Application object” on page 353

“Station object” on page 392

Description

Read only so cannot be set from script. Checks whether the operator has requested that the next display navigation be passed to the associated Station (as determined by the Flex Station Configuration Display properties).

Syntax

```
CrossScreenInvocationPending
```

Return value

True or *False*.

DecreaseZoomLevel method

Applicable to

"PlotCanvas object" on page 382

Description

Zooms out by approximately 20%.

Syntax

```
DecreaseZoomLevel
```

Returns values

None.

DeleteString method

Applicable to

"Combobox object" on page 363

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” on page 353

“Station object” on page 392

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.

Remarks

- This method is not available in eServer premium.

DPToLPX and DPToLPY methods

Applicable to

“AppWindow object” on page 359

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.

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” on page 353

“Station object” on page 392

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 'Command Reference' topic in the *Operator's 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” on page 353

“Station object” on page 392

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" on page 385

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.

Remarks

- After finding a popup/faceplate, you can access its properties, such as *close*, *top*, *bottom*, *left*, and *right*.

Example

This example closes the faceplate which displays 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” on page 350

“Checkbox object” on page 362

“Combobox object” on page 363

“Hyperlink object” on page 370

“Indicator object” on page 372

“Picture object” on page 380

“Pushbutton object” on page 386

“Shape object” on page 390

“Text object” on page 397

“Vector Graphic object” on page 400

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.

Example

```
Sub page_onpagecomplete
  'set focus to first alpha on page
  alpha1.focus
End Sub
```

FlagUserInteraction method

Applicable to

“Application object” on page 353

“Station object” on page 392

Description

Resets Station's user idle timeout.

Syntax

```
FlagUserInteraction
```

Return values

None.

GetActionIDForCommand method

Applicable to

“Application object” on page 353

Description

Returns the action ID of the specified command.

Syntax

```
GetActionIDForCommand commandname, actionID
```

Part	Description
commandname	The name of the command, as specified in Station's **Customize** dialog box.
actionID	The action ID assigned to the command.

Return values

None.

Example

This example, which is for a faceplate, overwrites Station's implementation of the 'Select SP' command to automatically select the SP element in the faceplate.

```
Sub Page_onexecutecommand
dim strCommand
  ' Get the command ID
 strCommand = window.event.getAttribute("onExecuteCommand", 0)
dim strAction
' Get the command's action ID
strAction = window.external.application.GetActionIDForCommand(strCommand)
' If the action is to Select the Set point then we'll handle it in script
  if strAction ="Select Setpoint" then
    alpha001.focus
    window.event.returnvalue = false 'Notify Station the command is handled
  end if
End Sub
```

GetBoundPlot method

Applicable to

“Plots object” on page 382

Description

Returns the point ID and parameter for a plot on a trend or basic trend.

Syntax

```
GetBoundPlot plotID, pointID, paramName
```

Part	Description
plotID	The plot's ID.
pointID	The ID of the point.
paramName	The point parameter.

Return values

The value of the point ID and parameter specified by the plot ID.

Example

This example shows how to get the point ID and parameter of a plot ID 1.

```
dim pointName, paramName
basictrend001.ChartArea.Plots.GetBoundPlot 1, pointName, paramName
```

GetConnectionProperty method

Applicable to

“Application object” on page 353

“Station object” on page 392

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" on page 390

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

- 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
```

Related topics

"Example: Using a shape custom property to store a user-defined value" on page 321
"Creating a dynamic shape" on page 71

GetDataTipArrowColors method

Applicable to

"EventBar object" on page 368

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" on page 387

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 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” on page 368

Description

Returns the visibility of data tips for an event.

Syntax

```
GetDataTipVisible 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

Tip

When using VBScript, True and False are predefined constants representing the values -1 and 0 respectively.

GetDeadband method

Applicable to

“ChartArea object” on page 361

Description

Returns the X Axis and Y Axis Deadband values. The Deadband is an area (in pixels) at the top, left, bottom, and right of the plot area that is left blank. The Deadband ensures the operator can see the full extent of the plots drawn. This method returns the number of pixels in the Deadband for each Axis.

Syntax

```
GetDeadband(pvarXAxisDeadband, pvarYAxisDeadband)
```

Part	Description
pvarXAxisDeadband	The Deadband value of X axis.
pvarYAxisDeadband	The Deadband value of Y axis.

Return Values

Deadband values of X Axis and Y Axis.

GetGridLinePen method

Applicable to

“PlotCanvas object” on page 382

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” on page 403

“YAxis object” on page 404

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” on page 361

“TitleToolbar object” on page 398

Description

Returns the visibility 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” on page 381

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” on page 387

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” on page 381

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 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" on page 399

"BasicTrend object" on page 360

Description

Returns the period of the trend.

Syntax

```
GetPeriod (nYears, nMonths, nWeeks, nDays, pHours, pMinutes, pSeconds)
```

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.
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” on page 374

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.

GetPlotDiscrete method

Applicable to

“Legacy object” on page 374

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” on page 374

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)
```

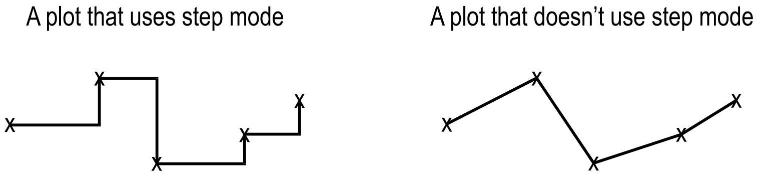
GetPlotStepMode method

Applicable to

“Legacy object” on page 374

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).



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” on page 374

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” on page 403

“YAxis object” on page 404

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" on page 374

"ReferenceCursor object" on page 387

Description

Returns an array of reference values and the corresponding dates/times for each plot intersected by the reference line.

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 *textbox009*) when the user creates a reference line (clicks in the plot view).

```
Sub trend001_OnReferenceCursorSet(nChart, bSet)
    on error resume next
    If bSet = 1 Then
        trend001.Legacy.GetReferenceValues plotIDs, refvaluess, refdateTimes
        textbox001.value = refdatetimes(0)
        textbox002.value = refvaluess(0)
        textbox003.value = refvaluess(1)
        textbox004.value = refvaluess(2)
        textbox005.value = refvaluess(3)
        textbox006.value = refvaluess(4)
        textbox007.value = refvaluess(5)
        textbox008.value = refvaluess(6)
        textbox009.value = refvaluess(7)
    End If
End Sub
```

GetResponse method

Applicable to

"Application object" on page 353

"Station object" on page 392

Description

Attention
This method is now deprecated. Use the GetResponse4 method.

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.

GetResponse2 method

Applicable to

"Application object" on page 353

Description

Attention
This method is now deprecated. Use the GetResponse4 method.

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.

GetResponse3 method

Applicable to

“Application object” on page 353

Description

Attention
This method is now deprecated. Use the GetResponse4 method.

Displays the specified prompt in Station's Message Zone and returns the user's response entered in Station's Command Zone. The OnResponse event fires when the user responds to the prompt.

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.

Related topics

“Using a button to start a pump” on page 575
“Adding point parameters to an object” on page 266

GetResponse4 method

Applicable to

“Application object” on page 353

Description

Displays the specified prompt in Station's Message Zone and returns the user's response. The OnResponse event fires when the user responds to the prompt.

Syntax

```
GetResponse4 prompt, type
```

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.

Return values

None.

Remarks

- This method is not available in eServer premium.
- The showCallout method may be an appropriate alternative to GetResponse4. This will display a callout beside the specified element rather than using the Message Zone. Error, prompt, and informational callouts are available.

Example

An operator is often required to print a trend which shows values that correspond to particular events occurring on site. To make sure the printouts are identifiable, the operator needs to add a description to the trend before printing it.

To achieve this, you create a custom display that contains the trend object and a button which uses the GetResponse4 method to prompt the operator to enter a description.

You attach the following script to the button's OnClick event:

```
Sub btnTrendDescription_onclick
  window.external.GetResponse4 "Enter trend description", 0
End Sub
```

This prompts the operator to enter a description. When the operator enters a description and clicks **OK**, the OnResponse event is fired.

In the OnResponse event, the event object is used to retrieve the response entered by the operator. The trend title is then set to the response, after which it can be printed by the operator.

```
Sub Page_onresponse
  strTrendDescription = window.event.getAttribute ("OnResponse", 0)
  trend001.TitleText = strTrendDescription
End Sub
```

Related topics

"showCallout method" on page 493

GetSampleInterval method

Applicable to

"Trend object" on page 399

"BasicTrend object" on page 360

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" on page 350

"Checkbox object" on page 362

"Combobox object" on page 363

"Group object" on page 369

"Indicator object" on page 372

"Page object" on page 378

"Picture object" on page 380

"Pushbutton object" on page 386

"Shape object" on page 390

"Text object" on page 397

"Vector Graphic object" on page 400

Description

Returns the value of a style attribute in the style associated with the object.

Syntax

```
GetStyleProperty styleattribute
```

Part	Description
styleattribute	The name of the style attribute.

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” on page 378

Description

Returns the value of a style attribute in a specific style.

Syntax

```
GetCustomProperty style, styleattribute
```

Part	Description
style	The name of the style.
styleattribute	The name of the style attribute.

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” on page 394

“DisplayDataRepository object” on page 366

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.

Related topics

“Example: Using a system custom property to store a user-defined value” on page 321
“Example: Using a shape custom property to store a user-defined value” on page 321
“Creating a standard display” on page 65

GetXAxisToolbarItemVisible method

Applicable to

“ChartArea object” on page 361

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” on page 382

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” on page 381

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 method

Applicable to

“Plots object” on page 382

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 method

Applicable to

“Plot object” on page 381

“Plots object” on page 382

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" on page 382

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.

HideAllDataTips method

Applicable to

"EventBar object" on page 368

Description

Hides all visible event data tips.

Syntax

```
HideAllDataTips
```

Return values

None.

HideDataTips method

Applicable to

"EventBar object" on page 368

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” on page 382

Description

Increases the zoom by approximately 20%.

Syntax

```
IncreaseZoomLevel
```

Return values

None.

InvokeCommand method

Applicable to

“Application object” on page 353

“Station object” on page 392

Description

Executes the specified Station command as if it had been selected by the user from a menu or toolbar. (It is equivalent to invoking the OnMenu2 event.)

Syntax

```
InvokeCommand command
```

Part	Description
command	The Station command that is executed. The command's name must be as specified in Station's **Customize** dialog box.

Return values

None.

Remarks

- If the display is called up in a browser (as opposed to being called up in Station), the result will be indeterminate.

Example

This example shows how to call up the point detail display (for the current point).

```
window.external.application.InvokeCommand "Request Point Detail"
```

InvokeMenu2 method

Applicable to

“Application object” on page 353

Description

Supersedes the 'InvokeMenu method' for scripts on DSP displays, which used a menu tag (number) instead of a command name.

Invokes a Station command, as if the user had chosen menu item or clicked toolbar button.

Syntax

```
InvokeMenu2 commandname
```

Part	Description
commandname	The name of the command, as specified in Station's **Customize** dialog box.

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.application.InvokeMenu2 "Request Point Detail"
```

InvokePopup method

Applicable to

“Application object” on page 353

“Station object” on page 392

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.InvokePopup "procedure5.htm", 100, 100, 0
```

InvokePopup2 method

Applicable to

"Application object" on page 353

"Station object" on page 392

Description

Calls up a popup or faceplate as if it had been called up by the user and docks the popup or faceplate so that it is visible if the user navigates to another display.

Syntax

`InvokePopup2 filename, Xpos, Ypos, popuptype, vbPersist`

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

Part	Description
vbPersist	Specifies whether the faceplate or popup is docked. *True* = The faceplate or popup is docked. False = The faceplate or popup is not docked.

Return Values

None.

Remarks

- This method is not available in eServer premium.
- In multi-window Station, the *vbPersist* parameter is not applicable.

Example

This example shows how to call up a popup called 'procedure5.htm' so that it is docked.

```
window.external.InvokePopup2 "procedure5.htm", 100, 100, 0, True
```

InvokeShortcutMenu method

Applicable to

"Popup object" on page 385

Description

Calls up a shortcut menu.

Syntax

```
InvokeShortcutMenu MenuFileName, Xpos, Ypos
```

Part	Description
MenuFileName	The filename (and relative path) of the shortcut menu. Note that the relative path is not required if the file is stored in the same Station display folder as the current display page, or in one of Station's other display folders.
Xpos *Ypos*	The x- and y-coordinates of the top-left of the menu relative to the parent display. (The coordinates of the top-left of the whole display page are *1 1*.)

Return values

None.

Example

This example calls up a shortcut menu called 'myShortcutMenu' (which is in the same Station display folder as the current display page) and places it with its top-left corner at the position of the mouse-click event on the display page object.

```
HDXPopupBehavior.InvokeShortcutMenu "myShortcutMenu.dita", window.event.x, window.event.y
```

Item method

Applicable to

“Dictionary object” on page 365

“StationWindows object” on page 395

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” on page 353

“Station object” on page 392

Description

Only available to Station Scripting Objects (SSOs). 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” on page 353

“Station object” on page 392

Description

Adds an entry in the Station log. The Station log, *log.txt*, is located in *<data folder>\Honeywell\HMIWebLog\*.

Where *<data folder>* is the location where Experion data is stored. For default installations, *<data folder>* is *C:\ProgramData*. The *C:\ProgramData* folder is a system folder, which means that it is only visible if you select the **Show hidden files, folders, and drives** option button in the **Folder Options** dialog box. To change this setting in Windows Explorer, click **Organize** > **Folder and search options**, and then click the **View** tab.

Syntax

```
LogMessage message
```

Part	Description
message	The message to be added to the Station log.

Return values

None.

Example

This example adds an entry to the log.

```
window.external.Parent.LogMessage "Operation complete"
```

LPToDPX and LPToDPY methods

Applicable to

"AppWindow object" on page 359

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).

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" on page 353

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" on page 392

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: • *0* = OK • *1* = OK and Cancel • *2* = Abort, Retry and Ignore • *3* = Yes, No and Cancel • *4* = Yes and No • *5* = Retry and Cancel

Return values

The code indicating which button was clicked:

1 = OK

2 = Cancel

3 = Abort

4 = Retry

5 = Ignore

6 = Yes

7 = No

Remarks

- Instead of using message boxes, Honeywell strongly recommends that you use temporary message zone text or callouts to display messages. Refer to the topic "Station object" for additional information about displaying temporary messages using window.external.TemporaryMessageZoneText. Refer to "showCallout method" for details about callouts.
- If you are creating a Station Scripting Object (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.

Related topics

“showCallout method” on page 493

“Station object” on page 392

Objects method

Applicable to

“Shape object” on page 390

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
```

Related topics

“Referencing objects” on page 301

“Event bubbling” on page 287

“Working with shapes” on page 303

“Shape object” on page 390

PutValue method

Applicable to

“StationDataRepository object” on page 394

“DisplayDataRepository object” on page 366

Description

Sets the value of a system custom property in the Station data repository.

Syntax

```
PutValue name, value
```

Part	Description
name	The name of the system custom property
value	The value to which the system custom property is set.

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"
```

Related topics

“Example: Using a system custom property to store a user-defined value” on page 321
“Example: Using a shape custom property to store a user-defined value” on page 321
“Creating a standard display” on page 65

Quit method

Applicable to

“Application object” on page 353

“Station object” on page 392

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" on page 353

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 event and can use the parameters that are passed to determine if the event is relevant to them.

Syntax

RaiseCustomEvent(*bstrURN*, *bstrNamespace*, *svarEventObject*)

Part	Description
bstrURN	The name of the event.
bstrNamespace	The name of the group of events to which this event belongs.
svarEventObject	The value of an object related to the event. The value can be of any form, such as text or numeric.

Remarks

- This method is not available in eServer premium.

Related topics

"Configuring Custom Events in Station" on page 324

Redraw method

Applicable to

“Legacy object” on page 374

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 scroll bar only *32* = X scroll bar 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” on page 353

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” on page 390

Description

Reloads the content of the shape if its autoReloadContent property is set to *`False`*.

Syntax

```
ReloadContent
```

Return values

None.

Remarks

- After using this 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.

Remove method

Applicable to

“Dictionary object” on page 365

“DisplayDataRepository object” on page 366

“StationDataRepository object” on page 394

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" on page 374

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” on page 374

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” on page 350

“Checkbox object” on page 362

“Combobox object” on page 363

“Hyperlink object” on page 370

“Indicator object” on page 372

“Picture object” on page 380

“Pushbutton object” on page 386

“Shape object” on page 390

“Text object” on page 397

“Vector Graphic object” on page 400

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*.)

Attention
You cannot write to RequestDemandData references because they are read-only.

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
```

Related topics

“Reading and writing to point parameters” on page 308

RequestServerLRN method

Applicable to

“Application object” on page 353

“Station object” on page 392

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 '*Server display program*' in the '*Customizing Stations*' section of the *Configuration Guide*.
- For more information about displaying the LRN assigned to an application program, see '*Starting a task automatically*' in the '*Controlling the execution of a Server API application*' section of 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" on page 353

"Station object" on page 392

Description

Requests the server task with the associated LRN (Logical Resource Number), with the specified parameters.

Syntax

```
RequestTask LRN [param1, param2, param3, param4, path]
```

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. **Attention** When calling the Server Display Program (LRN 21), these parameters are not optional. Unused parameters must contain a zero '0'.

Part	Description
path	Only applicable when calling the Server Display Program (LRN 21). Passes a string (like that which would normally be entered by the user directly in Station's Command Zone if the function was performed manually), for example like the name of a display or the name of a point. **Attention** Must be enclosed within quotes.

Return values

None.

Remarks

- This method is similar to RequestServerLRN method; however, it allows greater flexibility when scripted and 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 '*Server display program*' in the '*Customizing Stations*' section of the *Server and Client Configuration Guide*.
- For more information about displaying the LRN assigned to an application program, see '*Starting a task automatically*' in the '*Controlling the execution of a Server API application*' section of the *Application Development Guide*.

Examples

This example on a DSP display page uses the Server Display Program (LRN 21) to call up a different DSP display named 'testPageCallup.dsp:'

```
Application.RequestTask 21, 1, 0, 0, 0, "testPageCallup.dsp"
```

This example on a DSP display page uses the Server Display Program to call up an HMIWeb display named 'testPageCallup.htm:'

```
Application.RequestTask 21, 1, 0, 0, 0, "page://testPageCallup.htm"
```

These examples on an HMIWeb display page use the Server Display Program to call up an HMIWeb display named 'testPageCallup.htm:'

```
window.external.Parent.RequestTask 21, 1, 0, 0, 0, "page://testPageCallup.htm"
window.external.Parent.RequestTask 21, 1, 0, 0, 0, "testPageCallup.htm"
window.external.Parent.RequestTask 21, 1, 0, 0, 0, "testPageCallup"
```

This example on an HMIWeb display page uses the Server Display Program to call up the point detail display for a point named 'tank5:'

```
window.external.Parent.RequestTask 21, 73, 0, 0, 0, "tank5"
```

This example on an HMIWeb display page uses the Server Display Program to call up the trend display for a point named 'tank5:'

```
window.external.application.RequestTask 21, 8, 0, tank5, 0, ""
```

This example on an HMIWeb display page uses the Server Display Program to call up the point detail display listing for all points starting with the letter 'a:'

```
window.external.Parent.RequestTask 21, 73, 0, 0, 0, "a*"
```

ResetContent method

Applicable to

"Combobox object" on page 363

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" on page 399

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" on page 403

"YAxis object" on page 404

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" on page 353

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.

SetBoundPlot method

Applicable to

"Plots object" on page 382

Description

Adds a new plot or modifies an existing plot on a trend or basic trend.

Syntax

```
SetBoundPlot plotID, pointID, paramName
```

Part	Description
plotID	The plot's ID.
pointID	The ID of the point.
paramName	The point parameter.

Return values

None.

Example

This example shows how to set the point ID and parameter for plot ID 1.

```
basictrend001.ChartArea.Plots.SetBoundPlot 1, "sinewave", "pv"
```

Remarks

- This is the only method you can use to add a plot to a basic trend. For trends, you can also use the Legend object.

SetCommandUIState method

Applicable to

"Station object" on page 392

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 commandname, state
```

Part	Description
commandname	The name of the Station command, as specified in Station's **Customize** dialog box.
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" on page 390

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.
- The implementation of SetCustomProperty for shapes involves the re-creation of the shape object itself. This can lead to problems if SetCustomProperty is called from the OnUpdate event handler of a shape sub-element. To avoid this, in the OnUpdate event handler, use `window.setTimeout`, specifying a general function and a timer value of 1 ms. Then call the SetCustomProperty from within the interval function. This creates a slight delay during which the firing of update events for all elements on the page can be completed before the shape is re-created.
- 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.
- Use the GetCustomProperty method to get value of a custom property.

Example

```
Sub alpha001_onupdate

intTimer = window.setTimeout ("myfunction()", 1)

End Sub


Dim intTimerFunction myfunction()
window.clearTimeout(intTimer)
shape015.SetCustomProperty "point", "testproperty", "sinewave2"
End Function
```

Example scenarios

"Changing a trend's contents at run time - Scenario 1" on page 548

Related topics

"Example: Using a shape custom property to store a user-defined value" on page 321
"Creating a dynamic shape" on page 71

SetDataTipArrowColors method

Applicable to

"EventBar object" on page 368

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" on page 387

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” on page 368

Description

Sets the visibility of data tips for the specified event.

Syntax

```
SetDataTipVisible (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:

False = Hidden

True = Visible

Tip

When using VBScript, True and False are predefined constants representing the values –1 and 0 respectively.

SetDeadband method

Applicable to

“ChartArea object” on page 361

Description

Sets the XAxis and YAxis Deadband values. The Deadband is an area (in pixels) at the top, left, bottom and right of the plot area that is left blank. The Deadband ensures the operator can see the full extent of the plots drawn. This method defines the number of pixels that make up the Deadband.

Syntax

```
SetDeadband(nXAxisDeadband,nYAxisDeadband)
```

Part	Description
nXAxisDeadband	The Deadband value of XAxis to be set.
nYAxisDeadband	The Deadband value of YAxis to be set.

Return Values

None

SetGridLinePen method

Applicable to

"PlotCanvas object" on page 382

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" on page 403

"YAxis object" on page 404

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" on page 402

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.

Remarks

- The setInterval method continuously evaluates the specified expression until the timer is removed with the clearInterval method.

SetItemVisible method

Applicable to

"ChartToolbar object" on page 361

"TitleToolbar object" on page 398

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" on page 381

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” on page 387

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” on page 376

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" on page 381

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" on page 399

"BasicTrend object" on page 360

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" on page 374

Description

Sets the color of a plot.

Syntax

```
SetPlotColor plotID, linecolor
```

Part	Description
plotID	The plot's ID.
linecolor	The color of the plot.

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" on page 374

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” on page 374

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.

Part	Description
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" on page 374

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 **Attention** Line styles other than solid, are not supported if linewidth is greater than one pixel.
linewidth	The width of the plot line (in pixels).

Part	Description
linecolor	The color of the plot.

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" on page 374

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#
```

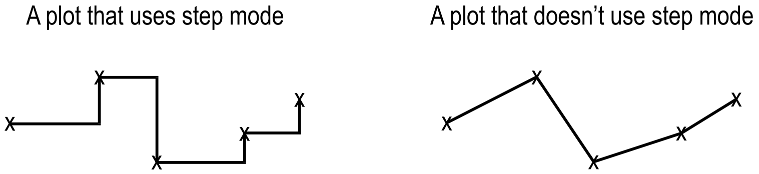
SetPlotStepMode method

Applicable to

“Legacy object” on page 374

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” on page 374

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” on page 399

“BasicTrend object” on page 360

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" on page 399

"BasicTrend object" on page 360

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" on page 402

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.

Remarks

- The specified expression or function is evaluated once.

SetTimeRange method

Applicable to

"Trend object" on page 399

"BasicTrend object" on page 360

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" on page 392

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" on page 392

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.

Remarks

- This method is not available in eServer premium.

SetXAxisToolbarItemVisible method

Applicable to

"ChartArea object" on page 361

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 • *6* = Remove reference line • *7* = Zoom/pause labels
bVisible	The visibility settings: • *0* = Hidden • *1* = Visible

Return values

None.

SetXAxisScale method

Applicable to

"Plots object" on page 382

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 time zones, 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” on page 381

“Plots object” on page 382

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” on page 382

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 time zones, 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" on page 381

"Plots object" on page 382

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" on page 382

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" on page 353

"Station object" on page 392

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
```

showCallout method

Applicable to

“Application object” on page 353

Description

Displays a callout beside the specified element. The callout type may be informational, error or prompt. If a user response is received the OnResponse or OnCalloutResponse event fires.

Syntax

Part	Description
element	The element on the display to be used to position the callout.
message	The text to be displayed.
type	The callout type: *0*:Presents the user with a prompt callout with **OK** and **Cancel** buttons. *1*: Presents the user with a prompt callout with **Yes** and **No** buttons. 2: Presents the user with an error type callout. 3: Presents the user with an informational type callout.
timeout	The number of seconds after which the callout will be automatically dismissed if it is still visible. Valid values are 0 (the callout will not auto-dismiss) and between 5 and 600. A value greater than 0 but less than 5 will default to 5.

Return values

None.

Remarks

- If callouts are disabled this method will operate via the Message Zone.
- Due to limitations involving custom events, the user response cannot be exposed as an attribute on the event object for the element-level OnCalloutResponse custom event. It is only available in this event as a property on Station’s Application object, "calloutresponse."
- User responses will be handled as follows:
  - If the element referenced by the callout defines the custom event OnCalloutResponse then that event handler will be triggered upon user response.
  - If the OnCalloutResponse event is not defined then the page-level OnResponse event will be triggered.
- Callouts may be dismissed before the end of the timeout period. This can occur in the following situations:
  - When a new display or faceplate is invoked.
  - When an action is performed that results in a new callout.
  - When the prompt is acknowledged (in the case of a prompt callout)
  - When the operator clicks away from the selected element. In this case there may be a delay as error callouts will only be dismissed after 5 seconds and prompt callouts will only be dismissed after 30 seconds (unless the timeout period is less than 30 seconds).
- If a prompt type callout is dismissed without the prompt being acknowledged then neither OnCalloutResponse nor OnResponse will be fired.

- Callouts are not supported within eServer. Scripts using callouts do not need to be adjusted for display using eServer, but callouts must be disabled on the eServer so that the method operates via the Message Zone instead. For more information, see the "General tab, server wide settings" topic in the *Server and Client Configuration Guide*.

Examples

The following examples illustrate the use of callouts invoked via script and the OnCalloutResponse (element-level) and OnResponse (page-level) events respectively. The first example illustrates the use of element-level event handlers through the scenario of an operator clicking a button, 'btnPrintTrendAlpha,' which in turn prints an associated trend.

To trigger a prompt type callout the following script is attached to the button’s OnClick event.

```
Sub btnPrintTrendAlpha_onclick
  window.external.showCallout me, "Are you sure you wish to print trend Alpha?", 1, 0
End Sub
```

This displays a prompt type callout with the supplied message. When the operator selects Yes or No, Station checks whether a definition of OnCalloutResponse exists relative to the "btnPrintTrendAlpha button," and if so, triggers it. The following script is attached to the button’s OnCalloutResponse event:

```
Sub btnPrintTrendAlpha_oncalloutresponse
  doPrint = window.external.calloutresponse
  if doPrint = ‘Y’
      printTrendAlpha()
End Sub
```

This event must be defined in order to distinguish between and correctly handle various callout response events, i.e. the display could potentially contain multiple buttons or associated trends which show a callout whose user response must be handled in a manner specific to the element from which it was fired. For example, if there was a second button, "btnPrintTrendBeta," on the display, it would define an OnCalloutResponse handler which would differ as shown:

```
Sub btnPrintTrendBeta_oncalloutresponse
  doPrint = window.external.calloutresponse
  if doPrint = ‘Y’
      printTrendBeta()
End Sub
```

The next example illustrates the use of the page-level OnResponse callout response event handler (this is the same event fired for GetResponse4). In this example there are multiple elements which, when clicked, show a confirmation prompt-type callout and navigate to the next display. The following script would be attached to each element’s OnClick event:

```
Sub elementXYZ_onclick
  window.external.showCallout me, “Are you sure you wish to navigate to the next display?", 1, 0
End Sub
```

The OnCalloutResponse relative to the elements that trigger the event would be intentionally left blank/ undefined and the following script would then be attached to the page’s OnResponse event handler.

```
Sub Page_onresponse
  doNavigate = window.external.calloutresponse
  if doNavigate = ‘Y’
      navigateToNextDisplay()
End Sub
```

Note that these scenarios are not mutually exclusive. A display could have a mixture of callouts that trigger only the OnCalloutResponse event handler on the element relative to which they were called and callouts that trigger only the OnResponse page-level event handler (when a specific OnCalloutResponse event handler is not defined).

Also note that getAttribute is also supported (as for GetResponse4). This is shown in the following example.

```
Sub Page_onresponse
  doNavigate = window.event.getAttribute ("OnResponse", 0)
  if doNavigate = ‘Y’
      navigateToNextDisplay()
End Sub
```

Related topics

“Displaying messages to operators” on page 298

“MessageBox method” on page 458

“Station object” on page 392

“GetResponse4 method” on page 443

UnZoom method

Applicable to

“PlotCanvas object” on page 382

“Legacy object” on page 374

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” on page 353

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.

Related topics

“OnActivate event” on page 497

“OnActivated event” on page 498

“OnAlarm event” on page 498

“OnAppStartup event” on page 498

“OnAppStatusUpdate event” on page 499

“OnBeforeConnect event” on page 499

“OnBeforeOKBKeyPressed event” on page 500

“OnBlink event” on page 501

“OnBlur event” on page 501

“OnCalloutResponse event” on page 502

“OnChange event” on page 503

“OnClick event” on page 504

“OnConnect event” on page 504

“OnContextMenu event” on page 505

“OnCustomEvent event” on page 505

“OnDataChange event” on page 507

“OnDblClick event” on page 508

“OnDeactivate event” on page 508

“OnDictionaryValueChanged event” on page 509

“OnDisconnect event” on page 510

“OnDisplayNotFound event” on page 510

“OnEventDataTipDisplayed event” on page 511

“OnExecuteCommand event” on page 511

“OnFocus event” on page 512

“OnKeyDown event” on page 513

“OnKeyPress event” on page 513

“OnKeyUp event” on page 514

“OnMenu event” on page 515

“OnMenu2 event” on page 515

“OnMouseDown event” on page 516

“OnMouseEnter event” on page 517

“OnMouseLeave event” on page 517

“OnMouseMove event” on page 518

“OnMouseOut event” on page 519

“OnMouseOver event” on page 519

“OnMouseUp event” on page 520

“OnOKBKeyPressed event” on page 520

“OnOperatorChange event” on page 521

“OnOperatorCommand event” on page 522

“OnOperatorCommand2 event” on page 522

OnActivate event

Applicable to

“Alphanumeric object” on page 350

“Checkbox object” on page 362

“Combobox object” on page 363

“Group object” on page 369

“Hyperlink object” on page 370

“Indicator object” on page 372

“Page object” on page 378

“Picture object” on page 380

“Pushbutton object” on page 386

“Shape object” on page 390

“Text object” on page 397

“Vector Graphic object” on page 400

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.

Related topics

“Event firing order” on page 289

OnActivated event

Applicable to

“Application object” on page 353

“Page object” on page 378

Description

Fires when Station becomes the active application.

OnAlarm event

Applicable to

“Application object” on page 353

“Page object” on page 378

“Station object” on page 392

Description

Fires when a new alarm appears in the Station's Alarm Zone.

OnAppStartup event

Applicable to

“Application object” on page 353

“Station object” on page 392

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” on page 353

“Page object” on page 378

“Station object” on page 392

Description

Fires when there is a change in status of one or more boxes in Station's Status Bar.

Syntax

For the Application and Station objects:

```
OnAppStatusUpdate(changedfields)
```

Part	Description
changedfields	An array, which specifies the status changes.

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.

Remarks

- This event is not available in eServer premium.

OnBeforeConnect event

Applicable to

“Application object” on page 353

“Page object” on page 378

“Station object” on page 392

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.
- This event is not available in eServer premium.

OnBeforeOKBKeyPressed event

Applicable to

“Application object” on page 353

“Station object” on page 392

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.
- This event is not available in eServer premium.

OnBlink event

Applicable to

“Application object” on page 353

“Page object” on page 378

“Station object” on page 392

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” on page 350

“Checkbox object” on page 362

“Combobox object” on page 363

“Group object” on page 369

“Hyperlink object” on page 370

“Indicator object” on page 372

“Page object” on page 378

“Picture object” on page 380

“Pushbutton object” on page 386

“Shape object” on page 390

“Text object” on page 397

“Vector Graphic object” on page 400

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.

Related topics

“Event firing order” on page 289

OnCalloutResponse event

Applicable to

“Application object” on page 353

Description

The OnCalloutResponse event may be fired when the user responds to a callout prompt that was generated by the showCallout method. If the element referenced by the callout defines the custom event OnCalloutResponse then that event handler will be triggered upon user response. If the OnCalloutResponse event is not defined then the page-level OnResponse event will be triggered.

Example

The following examples illustrate the use of callouts invoked via script and the OnCalloutResponse (element-level) event. The first example illustrates the use of element-level event handlers through the scenario of an operator clicking a button, ‘btnPrintTrendAlpha’, which in turn prints an associated trend.

To trigger a prompt type callout the following script is attached to the button’s OnClick event.

```
Sub btnPrintTrendAlpha_onclick
  window.external.showCallout me, "Are you sure you wish to print trend Alpha?", 1, 0
End Sub
```

This displays a prompt type callout with the supplied message. When the operator selects Yes or No, Station checks whether a definition of OnCalloutResponse exists relative to the “btnPrintTrendAlpha button,” and if so, triggers it. The following script is attached to the button’s OnCalloutResponse event:

```
Sub btnPrintTrendAlpha_oncalloutresponse
  doPrint = window.external.calloutresponse
  if doPrint = ‘Y’
      printTrendAlpha()
End Sub
```

This event must be defined (in this specific example) in order to distinguish between and correctly handle various callout response events, i.e. the display could potentially contain multiple buttons or associated trends which show a callout whose user response must be handled in a manner specific to the element from which it was fired. For example, if there was a second button, “btnPrintTrendBeta”, on the display, it would define an OnCalloutResponse handler which would differ as shown:

```
Sub btnPrintTrendBeta_oncalloutresponse
  doPrint = window.external.calloutresponse
  if doPrint = ‘Y’
      printTrendBeta()
End Sub
```

OnChange event

Applicable to

"Alphanumeric object" on page 350

"Checkbox object" on page 362

"Combobox object" on page 363

"Hyperlink object" on page 370

"Indicator object" on page 372

"Picture object" on page 380

"Pushbutton object" on page 386

"Shape object" on page 390

"Text object" on page 397

"Vector Graphic object" on page 400

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. This behaviour is ideal for checking new values against current values, allowing for data validation.
- The OnDataChange event fires when there is a change to the object's point-related parameters.
- The event does not bubble.

Example

```
Sub alpha001_onchange
'check for an increase in value
If alpha001.value > alpha001.currentValue Then
    'create a temp variable to store value difference
    Dim ValDif
    'store the value difference
    ValDif = alpha001.value - alpha001.currentValue
    'measure the value difference
    If ValDif > 10 Then
        'create a temp variable to store message
        Dim Msg
        'construct and store message
        Msg = "Can't increase temperature by "
        'append to message if too long for one line
        Msg = Msg & "more than 10 degrees at a time"
        'show message in Station message zone
        window.external.MessageZoneText = Msg
        'cancel the change
        alpha001.CancelChange
        'cancel the event
        window.event.returnValue = "false"
    End If
End If
End Sub
```

OnClick event

Applicable to

“Alphanumeric object” on page 350

“Checkbox object” on page 362

“Combobox object” on page 363

“Group object” on page 369

“Hyperlink object” on page 370

“Indicator object” on page 372

“Page object” on page 378

“Picture object” on page 380

“Pushbutton object” on page 386

“Shape object” on page 390

“Text object” on page 397

“Vector Graphic object” on page 400

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” on page 575

“Showing and hiding objects” on page 573

Related topics

“Event firing order” on page 289

OnConnect event

Applicable to

“Application object” on page 353

“Station object” on page 392

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
```

Related topics

"Using Station Scripting Objects" on page 330

OnContextMenu event

Applicable to

"Alphanumeric object" on page 350

"Checkbox object" on page 362

"Combobox object" on page 363

"Group object" on page 369

"Hyperlink object" on page 370

"Indicator object" on page 372

"Page object" on page 378

"Picture object" on page 380

"Pushbutton object" on page 386

"Shape object" on page 390

"Text object" on page 397

"Vector Graphic object" on page 400

Description

Standard DHTML event that fires when the user right-clicks the object.

Remarks

- If you invoke a shortcut menu with a script on the oncontextmenu event of a group, then you need to add the following line after calling InvokeShortcutMenu. If you don't do this, the default system shortcut menu will appear, overriding the menu bought up via a script.

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

OnCustomEvent event

Applicable to

"Application object" on page 353

"Page object" on page 378

"Station object" on page 392

Description

Fires when a 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 passed 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 event 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.

Parameter	Description
OnCustomEvent_EventObject	The value of an object passed by the event firer. The value can be of any form, such as text or numeric.

Remarks

- This event is not available in eServer premium.

Related topics

"Configuring Custom Events in Station" on page 324

OnDataChange event

Applicable to

"Alphanumeric object" on page 350

"Checkbox object" on page 362

"Combobox object" on page 363

"Group object" on page 369

"Hyperlink object" on page 370

"Indicator object" on page 372

"Picture object" on page 380

"Pushbutton object" on page 386

"Shape object" on page 390

"Text object" on page 397

"Vector Graphic object" on page 400

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.
- 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
```

Related topics

"Reading and writing to point parameters" on page 308
"Accessing the point mapped to a shape custom property" on page 316

OnDblClick event

Applicable to

"Alphanumeric object" on page 350

"Checkbox object" on page 362

"Combobox object" on page 363

"Group object" on page 369

"Hyperlink object" on page 370

"Indicator object" on page 372

"Page object" on page 378

"Picture object" on page 380

"Pushbutton object" on page 386

"Shape object" on page 390

"Text object" on page 397

"Vector Graphic object" on page 400

Description

Standard DHTML event that fires when the user clicks the left mouse button on the object.

Related topics

"Event firing order" on page 289

OnDeactivate event

Applicable to

"Alphanumeric object" on page 350

"Checkbox object" on page 362

"Combobox object" on page 363

"Group object" on page 369

"Hyperlink object" on page 370

"Indicator object" on page 372

"Page object" on page 378

"Picture object" on page 380

"Pushbutton object" on page 386

"Shape object" on page 390

"Text object" on page 397

"Vector Graphic object" on page 400

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.

Related topics

"Event firing order" on page 289

OnDictionaryValueChanged event

Applicable to

"Application object" on page 353

"Page object" on page 378

"Station object" on page 392

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” on page 353

“Station object” on page 392

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” on page 353

“Station object” on page 392

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 Station Scripting Objects (SSOs) and external applications.
- This event is not available in eServer premium.

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” on page 399

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

OnExecuteCommand event

Applicable to

“Application object” on page 353

“Page object” on page 378

Description

Fires when the user chooses a Station command—for example chooses a menu item or clicks a toolbar button.

Unlike the OnMenu2 event, you can cancel the command before it is executed.

Syntax

For the Application object:

`OnExecuteCommand (commandname, cancelcommand)`

For the Page object, you must use the Event object to explicitly retrieve the command:

```
strCommandID = window.event.getAttribute ("OnExecuteCommand", 0)
```

Part	Description
commandname	The name of the Station command, as specified in Station's **Customize** dialog box.
cancelcommand	Specifies whether the command is executed: 0 = the command is cancelled. -1 = the command is executed.

Remarks

- You must ensure that OnExecuteCommand event scripts do not invoke another command. (If they do, the OnExecuteCommand event will fire again and Station may end up in an endless loop.)
- If this event is associated with a page, you should use the page object's hasFocus property to handle this event when the page has windows focus (as shown in the example below).
- If you do not want to be able to cancel the command, use the OnMenu2event.

Example

This example, which is for a faceplate, overwrites Station's implementation of the 'Select SP' command to automatically select the SP element in the faceplate.

```
Sub Page_onexecutecommand
  if page.hasFocus = true then
    dim strCommand
    ' Get the command ID
    strCommand = window.event.getAttribute("onExecuteCommand", 0)
    dim strAction
    ' Get the command's action ID
    strAction = window.external.application.GetActionIDForCommand(strCommand)
    ' If the action is to Select the Set point then we'll handle it in script
    if strAction ="Select Setpoint" then
      alpha001.focus
      window.event.returnvalue = false 'Notify Station the command is handled
    end if
  end if
End Sub
```

OnFocus event

Applicable to

“Alphanumeric object” on page 350

“Checkbox object” on page 362

“Combobox object” on page 363

“Group object” on page 369

“Hyperlink object” on page 370

“Indicator object” on page 372

“Page object” on page 378

“Picture object” on page 380

“Pushbutton object” on page 386

“Shape object” on page 390

“Text object” on page 397

“Vector Graphic object” on page 400

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.
- Honeywell recommends that you don't use this method as it can interfere with the standard HMIWeb operation, and suggest that you use an OnActivate event or OnClick event instead.

Related topics

“Event firing order” on page 289

OnKeyDown event

Applicable to

“Alphanumeric object” on page 350

“Checkbox object” on page 362

“Combobox object” on page 363

“Group object” on page 369

“Hyperlink object” on page 370

“Indicator object” on page 372

“Page object” on page 378

“Picture object” on page 380

“Pushbutton object” on page 386

“Shape object” on page 390

“Text object” on page 397

“Vector Graphic object” on page 400

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.

Related topics

“Event firing order” on page 289

OnKeyPress event

Applicable to

“Alphanumeric object” on page 350

“Checkbox object” on page 362

“Combobox object” on page 363

“Group object” on page 369

“Hyperlink object” on page 370

“Indicator object” on page 372

“Page object” on page 378

“Picture object” on page 380

“Pushbutton object” on page 386

“Shape object” on page 390

“Text object” on page 397

“Vector Graphic object” on page 400

Description

Standard DHTML event that fires when the user presses an alphanumeric key.

Remarks

- The OnKeyPress event fires for the following keys:
  - Letters: A - Z (uppercase and lowercase)
  - Numerals: 0 - 9
  - Symbols: ! @ # $ % ^ & * ( ) _ - + = < [ ] { } ,. / ? \ | ' ' " ~
  - System: ESC, SPACEBAR, ENTER
- If you want to detect when the user presses any key, use the OnKeyDown event and test the event object.

Related topics

“Event firing order” on page 289

OnKeyUp event

Applicable to

“Alphanumeric object” on page 350

“Checkbox object” on page 362

“Combobox object” on page 363

“Group object” on page 369

“Hyperlink object” on page 370

“Indicator object” on page 372

“Page object” on page 378

“Picture object” on page 380

“Pushbutton object” on page 386

“Shape object” on page 390

“Text object” on page 397

“Vector Graphic object” on page 400

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.

Related topics

“Event firing order” on page 289

OnMenu event

Applicable to

“Application object” on page 353

“Station object” on page 392

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.

Remarks

- This event is not available in eServer premium.

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” on page 353

“Page object” on page 378

Description

Fires when the user chooses a Station command—for example chooses a menu item or clicks a toolbar button.

Syntax

For the Application object:

```
OnMenu2(commandname)
```

For the Page object, you must use the Event object to explicitly retrieve the command:

```
strCommandID = window.event.getAttribute ("OnMenu2", 0)
```

Part	Description
commandname	The name of the Station command, as specified in Station's **Customize** dialog box.

Remarks

- This event supersedes the OnMenu event.
- If you want to be able to cancel the command before it is executed, use the OnExecuteCommand event.
- If this event is associated with a faceplate, the faceplate needs to have focus.
- You can use the InvokeCommand method to simulate the event.
- If the display is called up in a browser (as opposed to being called up in Station), the command that is returned will be indeterminate.

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" on page 350

"Checkbox object" on page 362

"Combobox object" on page 363

"Group object" on page 369

"Hyperlink object" on page 370

"Indicator object" on page 372

"Page object" on page 378

"Picture object" on page 380

"Pushbutton object" on page 386

"Shape object" on page 390

"Text object" on page 397

"Vector Graphic object" on page 400

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.

Related topics

"Event firing order" on page 289

OnMouseEnter event

Applicable to

"Alphanumeric object" on page 350

"Checkbox object" on page 362

"Combobox object" on page 363

"Group object" on page 369

"Hyperlink object" on page 370

"Indicator object" on page 372

"Page object" on page 378

"Picture object" on page 380

"Pushbutton object" on page 386

"Shape object" on page 390

"Text object" on page 397

"Vector Graphic object" on page 400

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.

Related topics

"Event firing order" on page 289

OnMouseLeave event

Applicable to

"Alphanumeric object" on page 350

"Checkbox object" on page 362

"Combobox object" on page 363

"Group object" on page 369

"Hyperlink object" on page 370

"Indicator object" on page 372

"Page object" on page 378

"Picture object" on page 380

"Pushbutton object" on page 386

"Shape object" on page 390

"Text object" on page 397

"Vector Graphic object" on page 400

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.

Related topics

"Event firing order" on page 289

OnMouseMove event

Applicable to

"Alphanumeric object" on page 350

"Checkbox object" on page 362

"Combobox object" on page 363

"Group object" on page 369

"Hyperlink object" on page 370

"Indicator object" on page 372

"Page object" on page 378

"Picture object" on page 380

"Pushbutton object" on page 386

"Shape object" on page 390

"Text object" on page 397

"Vector Graphic object" on page 400

Description

Standard DHTML event that fires when the user moves the mouse over the object.

Remarks

- Does not bubble and cannot be canceled.

Related topics

"Event firing order" on page 289

OnMouseOut event

Applicable to

"Alphanumeric object" on page 350

"Checkbox object" on page 362

"Combobox object" on page 363

"Group object" on page 369

"Hyperlink object" on page 370

"Indicator object" on page 372

"Page object" on page 378

"Picture object" on page 380

"Pushbutton object" on page 386

"Shape object" on page 390

"Text object" on page 397

"Vector Graphic object" on page 400

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.

Related topics

"Event firing order" on page 289

OnMouseOver event

Applicable to

"Alphanumeric object" on page 350

"Checkbox object" on page 362

"Combobox object" on page 363

"Group object" on page 369

"Hyperlink object" on page 370

"Indicator object" on page 372

"Page object" on page 378

"Picture object" on page 380

"Pushbutton object" on page 386

"Shape object" on page 390

"Text object" on page 397

"Vector Graphic object" on page 400

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.

Related topics

“Event firing order” on page 289

OnMouseUp event

Applicable to

“Alphanumeric object” on page 350

“Checkbox object” on page 362

“Combobox object” on page 363

“Group object” on page 369

“Hyperlink object” on page 370

“Indicator object” on page 372

“Page object” on page 378

“Picture object” on page 380

“Pushbutton object” on page 386

“Shape object” on page 390

“Text object” on page 397

“Vector Graphic object” on page 400

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.

Related topics

“Event firing order” on page 289

OnOKBKeyPressed event

Applicable to

“Application object” on page 353

“Station object” on page 392

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 Station Scripting Objects (SSOs) and external applications.

OnOperatorChange event

Applicable to

“Application object” on page 353

“Page object” on page 378

“Station object” on page 392

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” on page 353

“Station object” on page 392

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.
- This event is not available in eServer premium.

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” on page 353

“Page object” on page 378

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 do not 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
```

Remarks

- This event is not available in eServer premium.

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” on page 378

“Shape object” on page 390

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
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

Key	Code
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
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

Key	Code
9	1088028673
10	1088094209
11	1100283905
12	1100349441
13	1100414977
14	1100480513
15	1100546049
16	1100611585
17	1100677121
18	1100742657
19	1100808193
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

Key	Code
SYST MENU	1102184449
PROC NETWK STATS	1104543745
COMM NETWK STATS	1104609281
AM STATS	1104674817
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

Related topics

“Event firing order” on page 289

OnPageComplete event

Applicable to

“Page object” on page 378

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 event 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
```

Related topics

"Event firing order" on page 289

OnPageLoad event

Applicable to

"Application object" on page 353

"Station object" on page 392

Description

Fires when a page is loaded but before all objects have received their first data update. Scripts may reference all objects on a display in this event.

OnPause event

Applicable to

"BasicTrend object" on page 360

"Trend object" on page 399

Description

Fires whenever updates to the trend are paused.

OnPeriodicUpdate event

Applicable to

"Application object" on page 353

"Page object" on page 378

"Station object" on page 392

Description

Fires when Station receives a periodic update from the server.

OnPlotDataBufferFull event

Applicable to

"Trend object" on page 399

Description

Fires when the plot's data buffer becomes full. Each plot can hold a maximum of 4,065 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" on page 399

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

onpropertychange event

Applicable to

"Text object" on page 397

Description

Standard DHTML event that fires when the property of an object changes.

Remarks

- Updating an object's value fires this event before the object's value is updated. For the new value of the current object use the innerHTML property.
- This event should not be used on objects with blinking text color as the event will fire each time the object text blinks.
- The property that caused this event to fire can be identified with Window.event.propertyName.
- The event does not bubble.

Dynamically changing the following properties will trigger this event.

Property	Triggering property value	Window.event.propertyName
disabled	True (false does not trigger)	disabled
id	Any	id
style.*	Any	style.*
styleClass	Any	ClassName
textColor	Any	style.color
textColorBlink	True (false does not trigger)	style.color
title	Any	title
value	Any	innerText

Example

```
Sub Employee_Name_onpropertychange

  If Strcomp(Window.event.propertyName,"InnerText", vbTextCompare)=0 Then
     FirstLast.value = Employee_Name.innerHTML + " " + Employee_Surname.value
  End If

End Sub
```

OnQuit event

Applicable to

“Application object” on page 353

“Station object” on page 392

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 event

Applicable to

“Trend object” on page 399

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

Remarks

Used to notify whenever the reference line has been created so that script which configures the ReferenceCursor object properties may be run if required.

Related events

“OnRequestReferenceDataTip event” on page 533

OnRequestAppropriateElement event

Applicable to

“Page object” on page 378 (only applicable to a faceplate)

Description

Fires when the appropriate element ID is required. If auto-selection is enabled, this element is automatically selected when the faceplate is called up.

Remarks

- For details about when this event fires, see 'Faceplate auto-selection' in the *Server and Client Configuration Guide*.

Example

```
Sub Page_onrequestappropriateelement
  page.mostAppropriateElement = "ConAlpSP"
End Sub
```

OnRequestCommandElement event

Applicable to

“Page object” on page 378 (only applicable to a faceplate)

Description

Fires when the command element ID is required. The command element describes the point being displayed in the faceplate. This element is used to provide context for all of the faceplate's selection-independent commands such as alarm acknowledge, callup group, and so on.

Remarks

- For details about when this event fires, see 'Faceplate auto-selection' in the *Server and Client Configuration Guide*.

Example

```
Sub Page_onrequestcommandelement
  page.commandElement = "ConAlpTagname"
End Sub
```

OnRequestDataTip event

Applicable to

"Trend object" on page 399

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" on page 399

Description

Fires when an event data tip is about to be displayed for an event when trends 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.

OnRequestModeElement event

Applicable to

“Page object” on page 378 (only applicable to a faceplate)

Description

Fires when the mode element ID is required.

Remarks

- For details about when this event fires, see 'Faceplate auto-selection' in the *Server and Client Configuration Guide*.

Example

```
Sub Page_onrequestmodeelement
  page.modeElement = "cboMode"
End Sub
```

OnRequestReferenceDataTip event

Applicable to

“Trend object” on page 399

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.

OnRequestSetPointElement event

Applicable to

“Page object” on page 378 (only applicable to a faceplate)

Description

Fires when the set point element ID is required.

Remarks

- For details about when this event fires, see 'Faceplate auto-selection' in the *Server and Client Configuration Guide*.

Example

```
Sub Page_onrequestsetpointelement
  page.setpointElement = "alphaSP"
End Sub
```

OnRequestOutPutElement event

Applicable to

“Page object” on page 378 (only applicable to a faceplate)

Description

Fires when the output element ID is required.

Remarks

- For details about when this event fires, see 'Faceplate auto-selection' in the *Server and Client Configuration Guide*.

Example

```
Sub Page_onrequestoutputelement
  page.outputElement = "alphaOP"
End Sub
```

OnResponse event

Applicable to

“Application object” on page 353

“Page object” on page 378

“Station object” on page 392

Description

Fires when the user responds to a prompt (which appears as a callout or in Station's Message Zone) that was generated by one of the following methods:

- showCallout
- GetResponse4
- GetResponse3
- GetResponse2

Syntax

For the Application and Station objects:

```
onresponse(response)
```

For the Page object, you use the Event object to explicitly retrieve the user's response. For example:

```
strResponse = window.event.getAttribute ("OnResponse", 0)
```

Part	Description
response	The user's response. If the method used the YesNo prompt type, the response string will be either *Y* for yes or *N* for no.

Remarks

- The GetResponse2 method and GetResponse3 method are now deprecated.
- Refer to "GetResponse4 method" and "showCallout method" for examples that use the OnResponse event.

OnResume event

Applicable to

“BasicTrend object” on page 360

“Trend object” on page 399

Description

Fires whenever live updates to the trend are resumed.

OnScrollXAxis event

Applicable to

“BasicTrend object” on page 360

“Trend object” on page 399

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

“BasicTrend object” on page 360

“Trend object” on page 399

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” on page 390

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.

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” on page 390

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.

OnSilenceIKB event

Applicable to

“Application object” on page 353

“Station object” on page 392

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” on page 353

“Page object” on page 378

“Station object” on page 392

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.
- This event is not available in eServer premium.

OnTimeSelectorPositionChanged event

Applicable to

“BasicTrend object” on page 360

“Trend object” on page 399

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

“BasicTrend object” on page 360

“Trend object” on page 399

Description

Fires when the trend has finished loading and has received data.

OnUnload event

Applicable to

“Page object” on page 378

Description

Fires when the page is unloaded.

Related topics

“Event firing order” on page 289

OnUpdate event

Applicable to

“Alphanumeric object” on page 350

“Checkbox object” on page 362

“Combobox object” on page 363

“Group object” on page 369

“Hyperlink object” on page 370

“Indicator object” on page 372

"Picture object" on page 380

"Pushbutton object" on page 386

"Shape object" on page 390

"Text object" on page 397

"Vector Graphic object" on page 400

Description

Fires when the value of the object is updated from the server.

Remarks

- The event does not bubble.

OnUserObjectNotify event

Applicable to

"Application object" on page 353

"Page object" on page 378

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" on page 399

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.

OnViewportChanged event

Applicable to

"Page object" on page 378

Description

Fires when all elements in the view have received a data update and can be scripted against. This event is the Pan and Zoom display equivalent of the OnPageComplete event. The OnPageComplete event does not fire for Pan and Zoom displays, as any elements outside the viewport when the display is first loaded will not receive a data update, and therefore may not be ready for scripting.

Fires multiple times – in addition to firing on initial page load, it will also fire whenever the viewport is changed and new data bound elements come into view. Not every element on the entire display will necessarily have received a data update and be ready to be scripted against, therefore the *page.IsElementInView* property can be used on a per element basis. For performance reasons, however, it is recommended that you move as much functionality as possible to individual *OnUpdate* event handlers.

Example

```
function Page_onviewportchanged
{
  // Cache the viewport if doing multiple checks
  var viewport = Page.getViewport();

  if (Page.isElementInView(element1, viewport)) {
    If (element1.value < 100) {
      element1.fillColor = ‘gray’;
    }
  }
  if (Page.isElementInView(element2, viewport)) {
    If (element2.value < 100) {
      element2.fillColor = ‘gray’;
    }
  }
}
```

Tip

This example is in javascript rather than vbscript.

OnZoom event

Applicable to

"Trend object" on page 399

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" on page 374

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" on page 374

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" on page 374

Description

Fires when the user zooms in on a specific part of the chart.

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" on page 374

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 event

Applicable to

“Trend object” on page 399

Description

Fires when the plot's data buffer becomes full. Each plot can hold a maximum of 4,065 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” on page 399

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” on page 399

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” on page 399

Description

Fires when the user creates a reference line.

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 text boxes (textbox001 to textbox*nnn*) 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

This section provides examples of 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.
- The display, *script_examples.htm*, contains example scripts. It is located in *<install folder>\Honeywell\Experion PKS\Client\HMIWeb Display Builder\Examples*, where *<install folder>* is the location where Experion is installed.

Related topics

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 and 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
    alpha002.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"
```

Changing a trend's contents at run time - Scenario 1

Scenario

A display includes a trend object for which the legend hidden, but you want operators to be able 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
```

Related topics

“Accessing the point mapped to a system custom property” on page 317

Changing a trend's contents at run time - Scenario 2

Scenario

You have several similar processes and you want an operator to be able to call up trend data for a particular process 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.

You write scripts for the buttons that specify which point parameters are assigned to the custom properties (and, therefore, to the chart).

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
 shape001.autoReloadContent = false
 'set the custom properties
 displayDataRepository.PutValue "Point1", "POINTANA1"
 displayDataRepository.PutValue "Param1", "PV"
 displayDataRepository.PutValue "Point2", "POINTANA2"
 displayDataRepository.PutValue "Param2", "PV"
    'apply the changes to the shape
 shape001.ReloadContent
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.)

Related topics

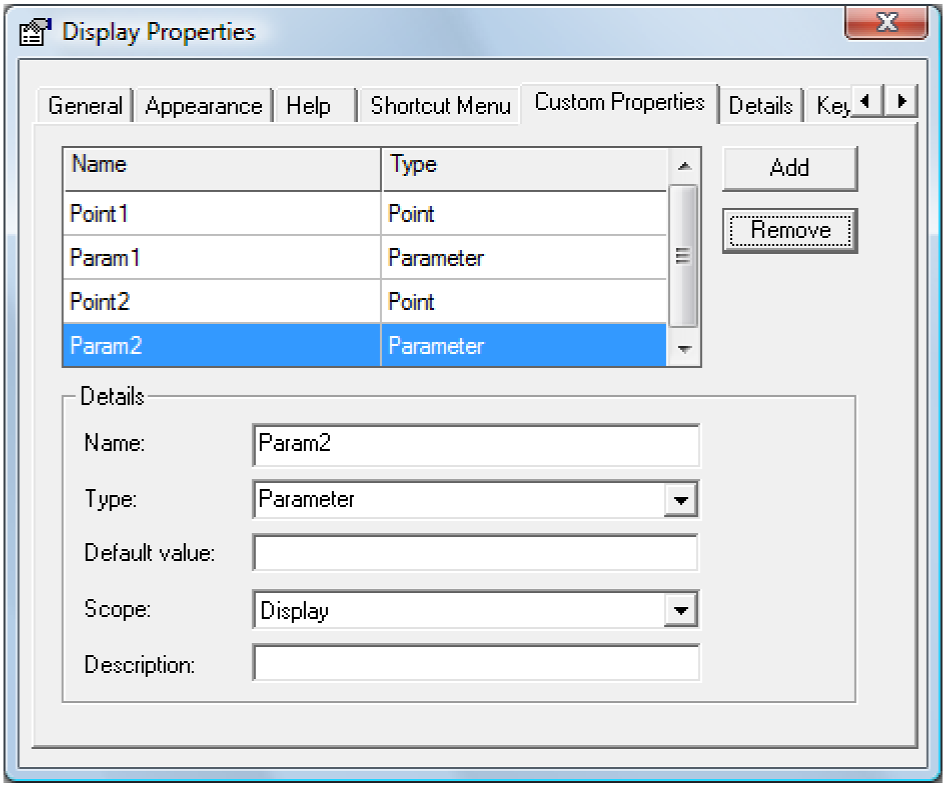
"Creating a display" on page 549

Creating a display

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.)



Name	Type
Point1	Point
Param1	Parameter
Point2	Point
Param2	Parameter

Details

Name: Param2

Type: Parameter

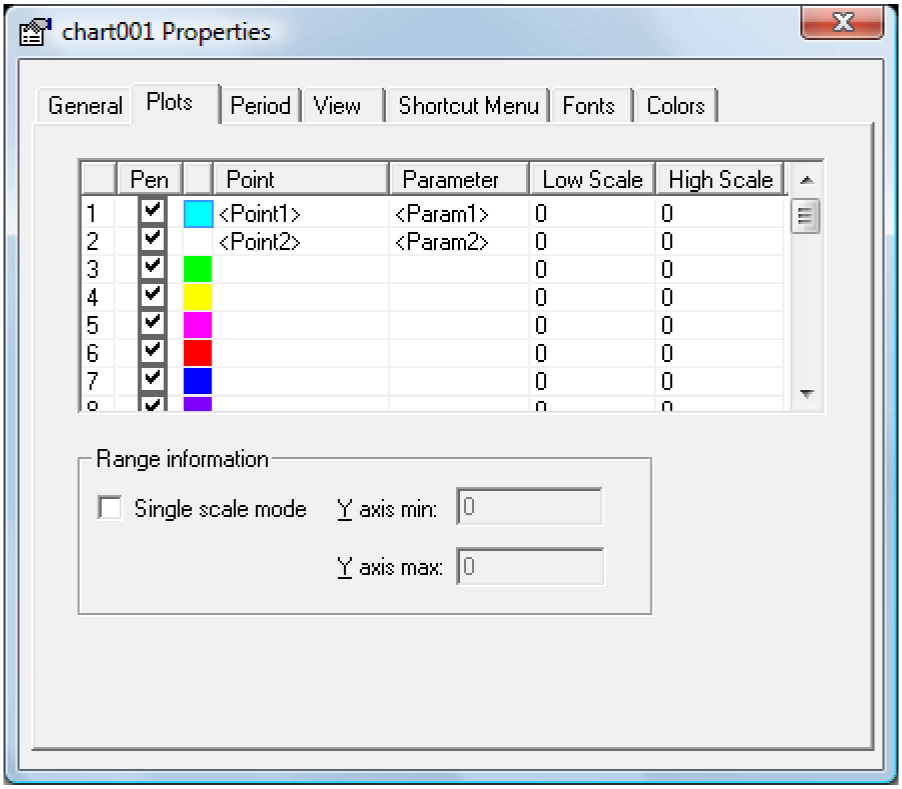
Default value:

Scope: Display

Description:

3 Add a trend to the display.

4 With the trend selected, click the Plots tab on the Properties Window and assign the custom properties to the plots as required.



	Pen		Point	Parameter	Low Scale	High Scale
1	☑	🟦	<Point1>	<Param1>	0	0
2	☑		<Point2>	<Param2>	0	0
3	☑	🟩			0	0
4	☑	🟨			0	0
5	☑	🟪			0	0
6	☑	🟥			0	0
7	☑	🟦			0	0
8	☑	🟪			0	0

Range information

□ Single scale mode   Y axis min: 0

Y axis max: 0

5 Configure the trend's other properties as appropriate, such as the number of samples, units and so on.

Related topics

“Changing a trend's contents at run time - Scenario 2” on page 549

Checking if an element is Not a Number (NaN)

Scenario

You need to check if the value of a point parameter is NaN.

Solution

Write a script using the helper methods provided.

Scripts

For a display element such as an alphanumeric:

```
If alpha001.IsElementNaN = true then
   'Code to execute if NaN
   .
   .
Else
   'Code to execute otherwise
   .
   .
End If
```

For a script data reference:

```
If rect001.IsNaN("point.parameter") = true then
   'Code to execute if NaN
   .
   .
Else
   'Code to execute otherwise
   .
   .
End If
```

Remarks

- It is possible to do a string comparison directly on the Value property (or DataValue property for script data), but the above helper methods are preferred as they will continue to give correct results even if the string representation of NaN changes.

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'.

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 flowcharts 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
```

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, 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.

Related topics

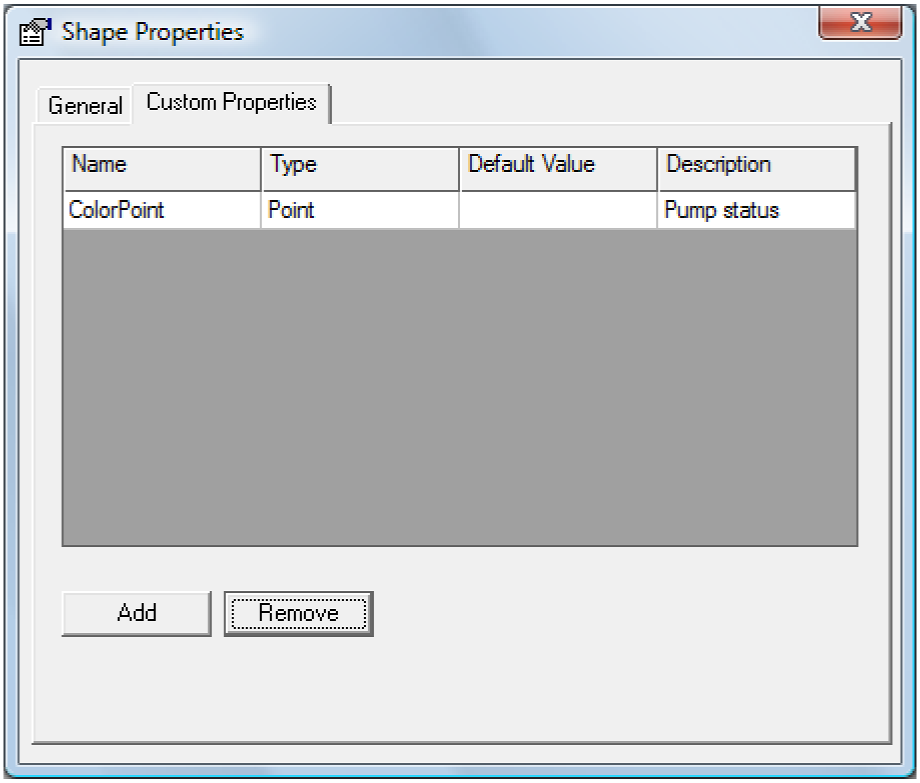
"Creating the dynamic shape" on page 555

"Inserting the shape into displays" on page 556

Creating the dynamic shape

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`.

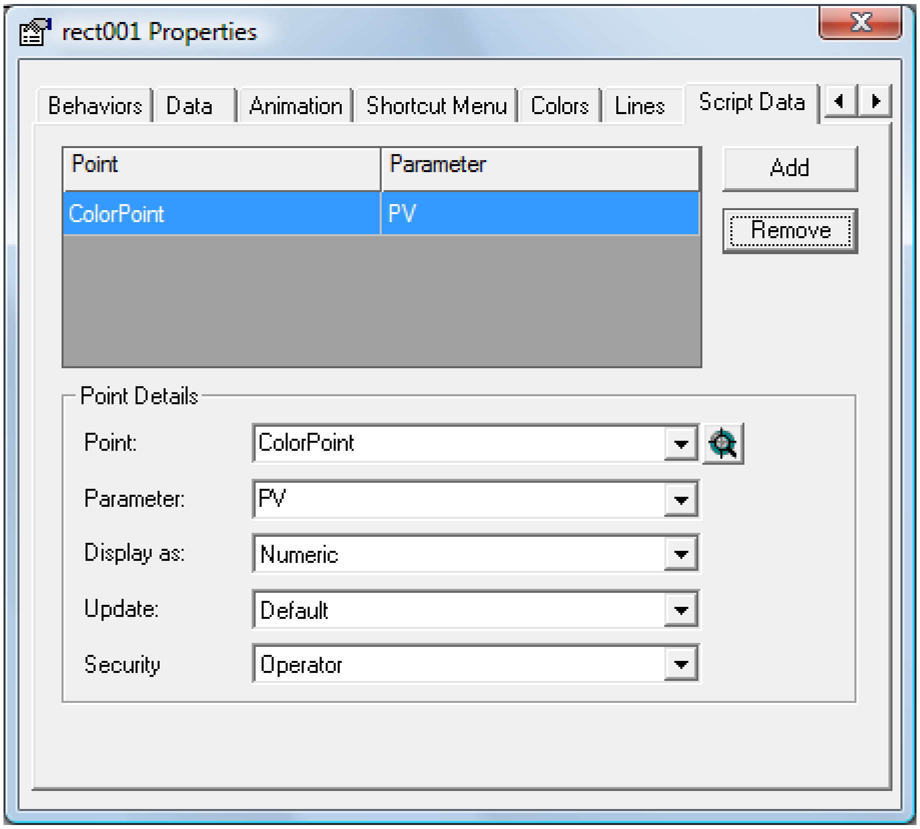


Name	Type	Default Value	Description
ColorPoint	Point		Pump status

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 a previous step as a custom property.



Point	Parameter
ColorPoint	PV

Point: ColorPoint
Parameter: PV
Display as: Numeric
Update: Default
Security: Operator

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
```

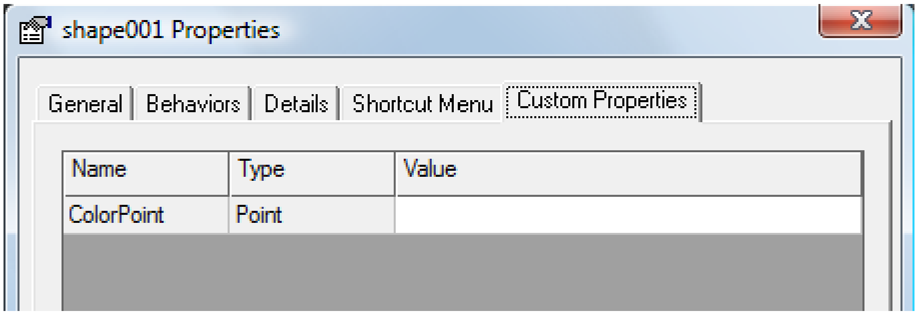
Related topics

"Inserting the shape into displays" on page 556

"Controlling the color of an object in a dynamic shape" on page 555

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**.



Name	Type	Value
ColorPoint	Point	

Related topics

“Creating the dynamic shape” on page 555

“Controlling the color of an object in a dynamic shape” on page 555

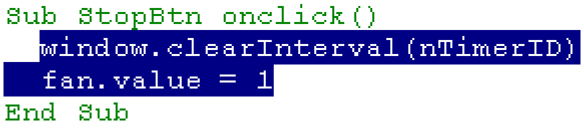
Copying example scripts

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.



```
Sub StopBtn onclick()
  window.clearInterval(nTimerID)
  fan.value = 1
End Sub
```

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.

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.

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
```

Related topics

“Reading and writing to point parameters” on page 308

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 combo box is used to issue the command.)

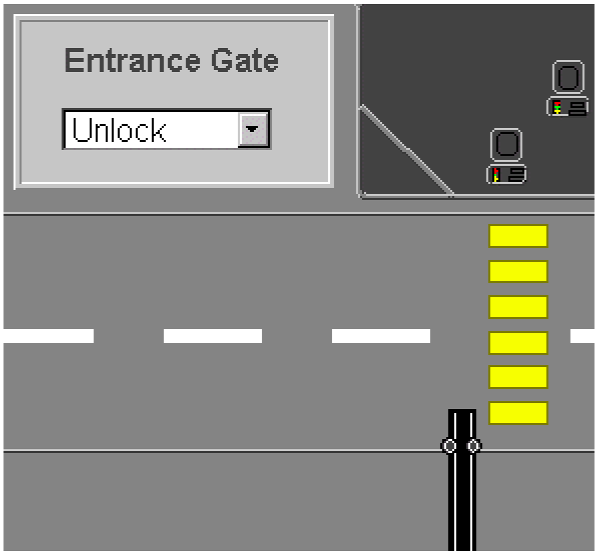


Figure 29: The gate in its open position

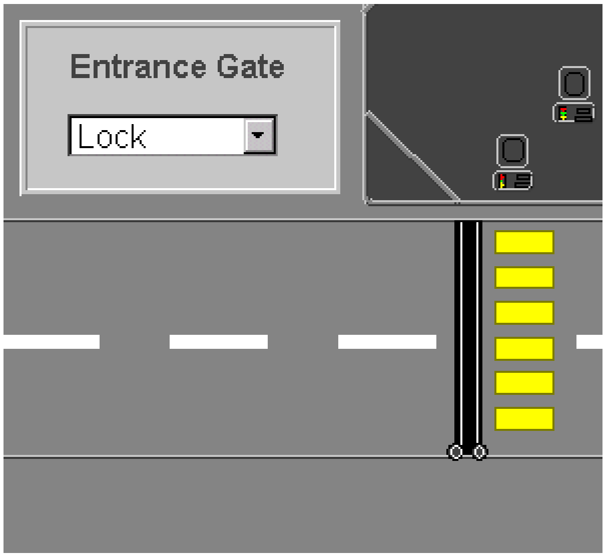


Figure 30: The gate in its closed position

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 combo box'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 combo box's onchange event. This creates a 100 ms timer, which controls the speed of the animation.

Based on the current state of the combo box, 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
```

Related topics

"Timers" on page 299

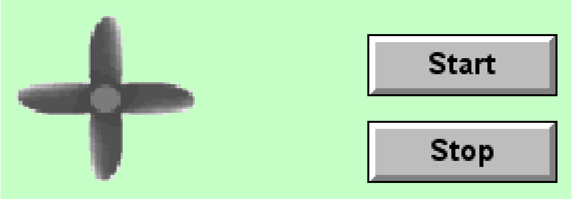
"General section" on page 297

Creating an animation with a shape sequence

Scenario

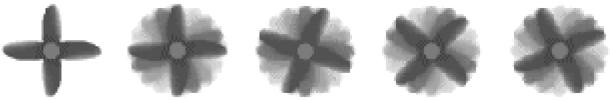
You have a fan that is controlled by **Start** and **Stop** buttons on a display, and want to animate the fan.

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.

Tip

Animations do not run in HMIWeb Display Builder. They can be viewed only in Station.

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.

Related topics

“Timers” on page 299
“General section” on page 297

Creating a 'Once-off scan' button for dynamic scanning

This topic shows you how to create a push button that, when clicked, performs a 'once-off' scan of all parameters that have a non-zero scan period, on a specified controller. You would typically use this to look at configuration values on a controller. Another use might be when an operator, concerned about the freshness of the data on exception-based controllers, wants a live update from the controller rather than rely on exception-based packets.

See *Scenario #4* in the topic titled "Dynamic scanning scenarios" in the *Server and Client Configuration Guide* for more information.

Prerequisites

- You have created a custom display and added alphanumeric objects with the parameters you want to scan on the controller.
- The **Properties** tab in HMIWeb Display Builder is visible (CTRL + R).

To create a 'Once-off scan' button for dynamic scanning

1. Click the ▬ **Pushbutton** icon in the toolbar.
2. Drag the pointer diagonally over the display, and release the mouse button when the dotted 'outline rectangle' is the correct size and shape.

   A new button appears, with **Button** as the label.
3. In the **Button Details** group of the **Properties** tab, click the **Label** row and type `Once-off scan`.

   The button label changes to **Once-off scan**.
4. In the **Behaviors** group of the **Properties** tab, click ... on the **Script Data** row.

   The **pushbutton*nnn* Properties** dialog appears. For example, **pushbutton001 Properties** or **pushbutton002 Properties**.

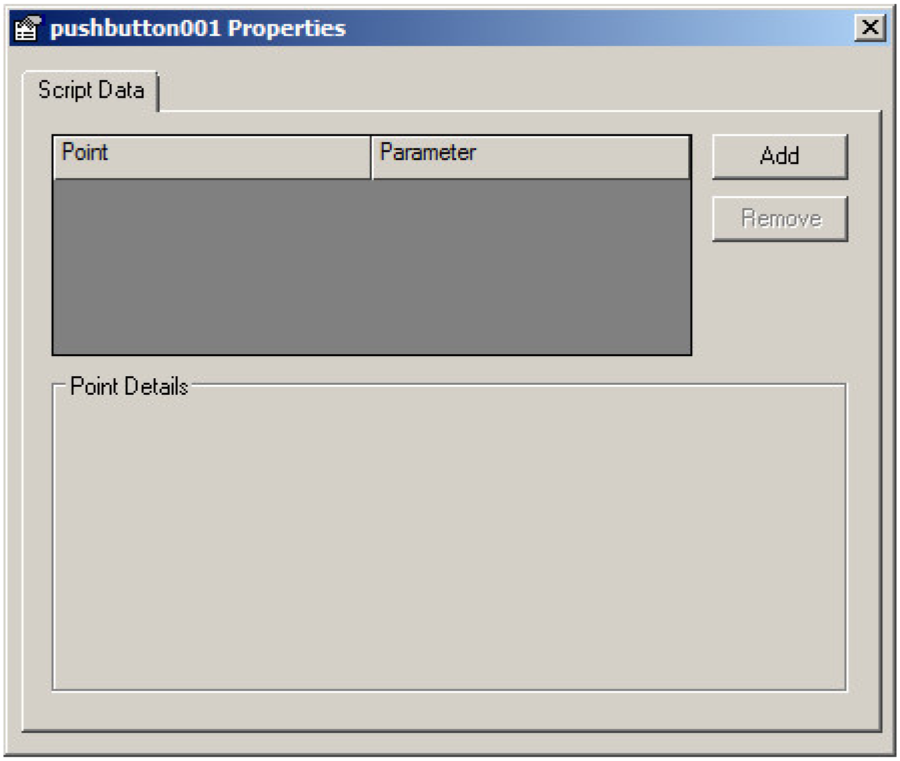


Figure 31: pushbutton*nnn* Properties dialog

5 Click **Add**.

A row appears with *`PV`* as the default parameter.

6 In the **Point** box, type the address of the controller in the format:

`$CONTROLLER`*`nnnn`*

where *`nnnn`* is the controller number. If you are addressing a controller on a DSA-connected server, use the pre-pended point name format. That is:

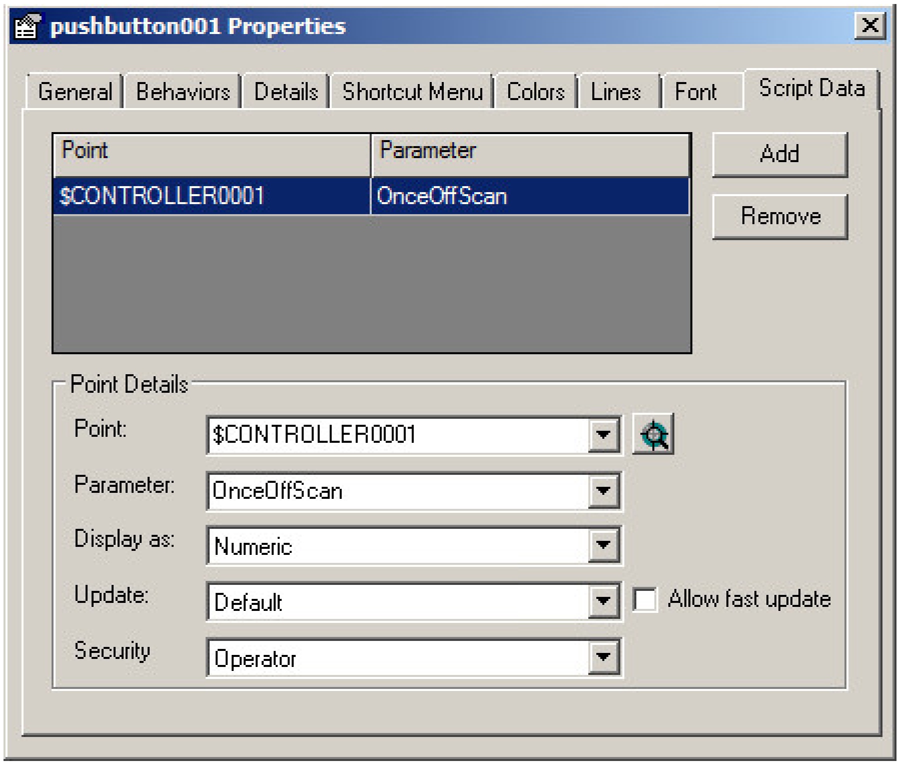
`<ServerAlias>``:$CONTROLLER`*`nnnn`*

7 In the **Parameter** box, enter or type `OnceOffScan`.

8 In the **Display as** box, select *`Numeric`*.

9 In the **Update** box, select *`Default`*.

10 In the **Security** box, select the appropriate security level.



Point	Parameter
$CONTROLLER0001	OnceOffScan

Point: $CONTROLLER0001

Parameter: OnceOffScan

Display as: Numeric

Update: Default ☐ Allow fast update

Security Operator

Figure 32: pushbutton*nnn* Properties dialog – OnceOffScan parameter

11 Close the **pushbutton***nnn* **Properties** dialog.

12 Right-click the pushbutton object and select **Edit script**, or press CTRL + ENTER.

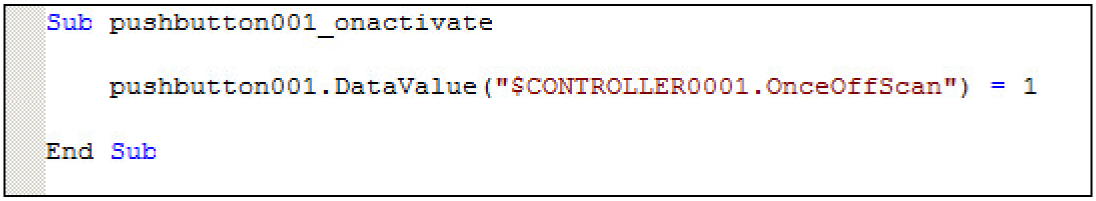
The **Script Editor** tab appears.

13 Type the following entry after the `Sub` row and before the `End Sub` row:

`pushbutton`*`nnn`*`.DataValue("$CONTROLLER`*`nnnn`*`.OnceOffScan") = 1`

where *`nnn`* is the push button number and *`nnnn`* is the controller number.

For example, `pushbutton001.DataValue("$CONTROLLER0001.OnceOffScan") = 1`



```
Sub pushbutton001_onactivate

    pushbutton001.DataValue("$CONTROLLER0001.OnceOffScan") = 1

End Sub
```

Figure 33: Script Editor example

14 Make other changes to the push button as necessary.

15 Save the display.

Creating a 'Time-limited dynamic scan' check box for dynamic scanning

This topic shows you how to create a check box to place on a display that, when selected, performs a periodic scan or accelerated scan (depending on configuration), for a period of two minutes, of all parameters that have a non-zero scan period, on a specified controller. Once the two-minute period expires, the scan rate returns the configured scan period (CSP), if one exists, and the check box is automatically cleared.

See *Scenario #5* in the topic titled "Dynamic scanning scenarios" in the *Server and Client Configuration Guide* for more information.

Prerequisites

- You have created a custom display and added alphanumeric objects with the parameters you want to scan on the controller.
- The **Properties** tab in HMIWeb Display Builder is visible (CTRL + R).

To create a 'Time-limited dynamic scan' check box for dynamic scanning

1. Click the ☑ **check box** icon in the toolbar.
2. Drag the pointer diagonally over the display, and release the mouse button when the dotted 'outline rectangle' is the correct size and shape.

   A new check box appears.
3. In the **Data** group of the **Properties** tab, on the **Point** row, type the address of the controller in the format:

   `$CONTROLLER`*`nnnn`*

   where *`nnnn`* is the controller number. If you are addressing a controller on a DSA-connected server, use the pre-pended point name format. That is:

   `<ServerAlias>``:$CONTROLLER`*`nnnn`*

   You can also click the down arrow next to the **Point** row and select the controller from the list.
4. In the **Parameter** box, enter or type `FastScan`.
5. Make other changes to the check box as necessary.
6. Save the display.

Forcing a display to use a particular SafeView window

Scenario

Attention
This example is only applicable if you use Station in conjunction with SafeView

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.

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.

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.

Related topics

“Preparing the ActiveX control” on page 570

Preparing the ActiveX control

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.

Related topics

“Responding to Station events from another application” on page 570

“ActiveX Script” on page 570

ActiveX Script

Scripts

The code for the general section of the class module for steps 3, 4 and 6 in Preparing the ActiveX control 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()
```

Related topics

"Preparing the ActiveX control" on page 570

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 sent. For more information about aliases, see the section 'Configuring Console Stations and Consoles' in the *Configuration Guide*.

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
```

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
```

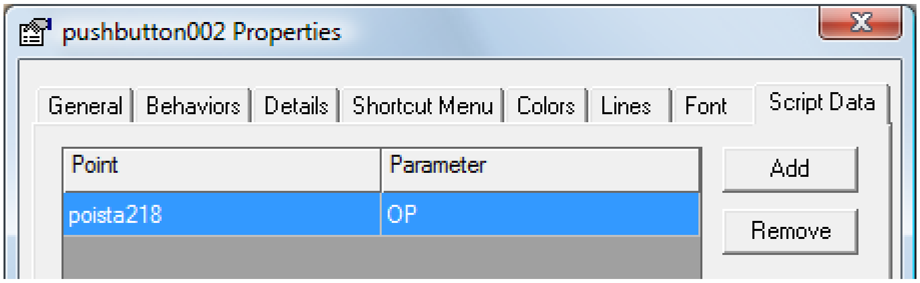
Using a button to start a pump

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'.



Point	Parameter
poista218	OP

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
```

Related topics

"Reading and writing to point parameters" on page 308

"Accessing the point mapped to a shape custom property" on page 316

"GetResponse3 method" on page 443

"Adding point parameters to an object" on page 266

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.

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
```

Related topics

“Reading and writing to point parameters” on page 308

Using event bubbling to reduce script effort

Scenario 1

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
```

Scenario 2

In the situation where several display elements co-exist on the same display page, each with its own element scripting, such as with animation, it is more efficient to create and utilise a common event handler, using a page level boolean variable to flag when it's time to start the page animation scripting.

Solution

The use of the subroutine *AlmTypeCheck* in the analogue system faceplates is an example of how the concept of page level common scripting is implemented.

Scripts

In this example, a global variable *bPageComplete* is defined and initialized as false. Whenever *AlmTypeCheck* is run, it first checks *bPageComplete* and only continues to run if it has been set to true. For example:

```
<SCRIPT language=VBScript defer>
  dim bPageComplete
  bPageComplete = false

  sub AlmTypeCheck()
  if bPageComplete = false then
    exit sub
  end if

    'Code to check
 'alarm type here.

end sub
</SCRIPT>
```

AlmTypeCheck is called in the OnUpdate event handler for various elements on the page:

```
<SCRIPT language=VBScript event=onupdate for=element defer>
  on Error resume next
  Call AlmTypeCheck()
</SCRIPT>
```

The variable *bPageComplete* is only set to true when the OnPageComplete event is handled.

```
<SCRIPT language=VBScript event=onpagecomplete for=Page defer>
  on error resume next
  bPageComplete = true
  AlmTypeCheck()
</SCRIPT>
```

Therefore the main body of code in *AlmTypeCheck* (not shown) will not be run until the page is complete despite the subroutine potentially being called numerous times before that point. Note that the OnPageComplete event handler calls *AlmTypeCheck* to ensure that it is run once as soon as the page has completed loading.

Related topics

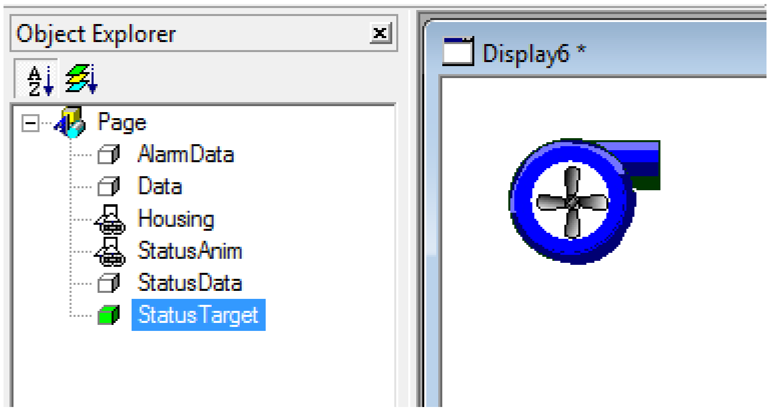
"Event bubbling" on page 287

"Style object" on page 396

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
```

Notices

Trademarks

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

OneWireless™ 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.

Third-party licenses

This product may contain or be derived from materials, including software, of third parties. The third party materials may be subject to licenses, notices, restrictions and obligations imposed by the licensor. The licenses, notices, restrictions and obligations, if any, may be found in the materials accompanying the product, in the documents or files accompanying such third party materials, in a file named third_party_licenses on the media containing the product, or at http://www.honeywell.com/ps/thirdpartylicenses.

Documentation feedback

You can find the most up-to-date documents on the Honeywell Process Solutions support website at:

http://www.honeywellprocess.com/support

If you have comments about Honeywell Process Solutions documentation, send your feedback to:

hpsdocs@honeywell.com

Use this email address to provide feedback, or to report errors and omissions in the documentation. For immediate help with a technical problem, contact your local Honeywell Process Solutions Customer Contact Center (CCC) or Honeywell Technical Assistance Center (TAC) listed in the “Support and other contacts” section of this document.

How to report a security vulnerability

For the purpose of submission, a security vulnerability is defined as a software defect or weakness that can be exploited to reduce the operational or security capabilities of the software.

Honeywell investigates all reports of security vulnerabilities affecting Honeywell products and services.

To report a potential security vulnerability against any Honeywell product, please follow the instructions at:

https://honeywell.com/pages/vulnerabilityreporting.aspx

Submit the requested information to Honeywell using one of the following methods:

- Send an email to security@honeywell.com.

  or

- Contact your local Honeywell Process Solutions Customer Contact Center (CCC) or Honeywell Technical Assistance Center (TAC) listed in the “Support and other contacts” section of this document.

Support

For support, contact your local Honeywell Process Solutions Customer Contact Center (CCC). To find your local CCC visit the website, https://www.honeywellprocess.com/en-US/contact-us/customer-support-contacts/Pages/default.aspx.

Training classes

Honeywell holds technical training classes on Experion PKS. 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.

Index