

# **Xycom Automation™ OpenHMI™**

## **User's Guide**

Revision A  
January, 1999

**Xycom Automation, Inc.**

<i>Revision</i>	<i>Description</i>	<i>Date</i>
A	Manual Released	1/99

***Trademark Information***

Xycom Automation and OpenHMI are trademarks of Xycom Automation, Inc.

Brand or product names are registered trademarks of their respective owners.

Windows is a registered trademark of Microsoft Corp. in the United States and other countries.

Wonderware FactorySuite, InTouch, WindowMaker, WindowViewer, SQL Access Manager, Recipe Manager, SPC Pro, DBDump, DBLoadd, HDMerge, HistData, Wonderware Logger, InControl, InTrack, InBatch, IndustrialSQL, FactoryOffice, Scout, SuiteLink and NetDDE are trademarks of Wonderware Corporation.

***Copyright Information***

This document is copyrighted by Xycom Automation Incorporated (Xycom Automation) and shall not be reproduced or copied without expressed written authorization from Xycom Automation.

The information contained within this document is subject to change without notice. Xycom Automation does not guarantee the accuracy of the information.

**Xycom Automation, Inc.**  
**750 North Maple Road**  
**Saline, MI 48176-1292**  
**734-429-4971 (phone)**  
**734-429-1010 (fax)**

---

# Table Of Contents

<b>WINDOWMAKER PROGRAM ELEMENTS</b>	<b>1-1</b>
<b>USING WINDOWMAKER</b>	<b>2-1</b>
<b>TAGNAME DICTIONARY</b>	<b>3-1</b>
<b>CREATING ANIMATION LINKS</b>	<b>4-1</b>
<b>CREATING QUICKSCRIPTS IN OPENHMI</b>	<b>5-1</b>
<b>ALARMS/EVENTS</b>	<b>6-1</b>
<b>TREND GRAPHS</b>	<b>7-1</b>
<b>I/O COMMUNICATIONS</b>	<b>8-1</b>

## CHAPTER 1

# WindowMaker Program Elements

WindowMaker is the development environment for OpenHMI. The WindowMaker graphical user interface adheres to Windows 95 and Windows NT GUI standards. WindowMaker supports floating and docking toolbars, right-mouse click menus throughout for quick access to frequently used commands and a customizable color palette that provides 16.7 million color support. (The color support is limited only by your video card capability.)

WindowMaker's Application Explorer provides you with a powerful, graphical method for navigating and configuring your OpenHMI applications. It provides you with easy access to WindowMaker's most commonly used commands and functions such as, all windows commands, all configuration commands and all OpenHMI QuickScript editors. Additionally, the Application Explorer will display all installed add-on programs such as Recipe Manager, and it provides you with a customizable application launcher.

On the Windows NT operating system, you can configure the Application Explorer to launch any other program to quickly switch between HMI configuration and I/O Server configuration.

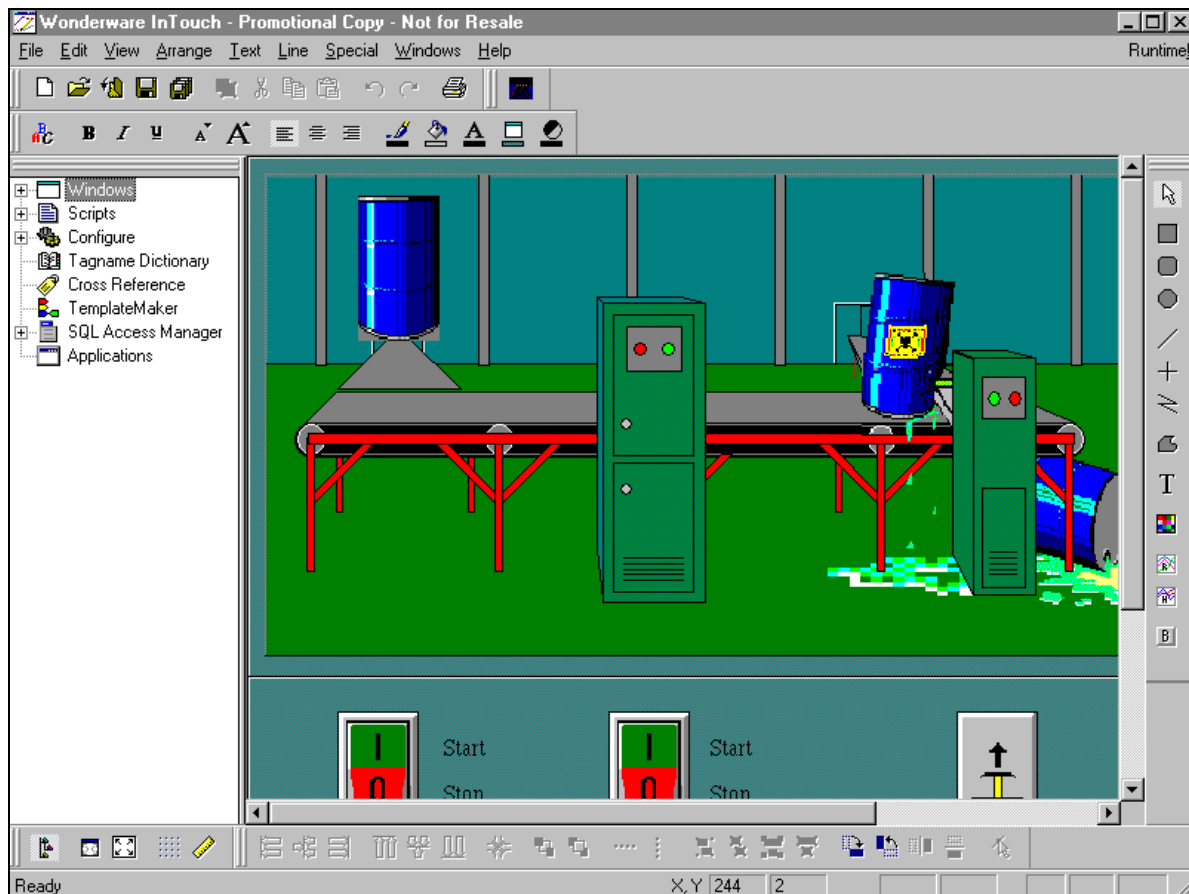
## The WindowMaker GUI

WindowMaker supports the Windows 95 and Windows NT operation systems graphic user interface (GUI) standards including, right-click mouse support, floating and docking toolbars, pull down menus, context-sensitive help and so on.

The WindowMaker development environment is configurable. By default when you initially open WindowMaker, most of the available elements are automatically displayed including, all toolbars, the Application Explorer and the status bar. However, you can show or hide any or all of these elements and, you can move the toolbars and the Application Explorer to any location that you desire within the WindowMaker window. You can also display the optional ruler and you can turn on and off the visible grid in your windows.

☞ For more information on moving the toolbars see, "Working with the Floating/Docking Toolbars."

The following illustrates the elements of the WindowMaker development environment:



When you create a new application, and run WindowMaker for the first time, its program elements will automatically appear in the default configuration shown in the illustration above.

☞ Many of the tools will not become active until a window is opened and objects are placed in the window and then selected. When a tool is not active, its functionality is not applicable for the current state of the window or the selected object.

---

**Note** When you close WindowMaker, the toolbar floating or docked positions and sizes, Application Explorer and, WindowMaker window size preferences are all saved. When you subsequently run WindowMaker they will be persistent.

---

## The Application Explorer

WindowMaker's Application Explorer is a hierarchical graphical view of your application. It shows you what items you have configured in your application and provides you easy access to those items. It also provides you with quick access to many of WindowMaker's most commonly used commands and functions.

---

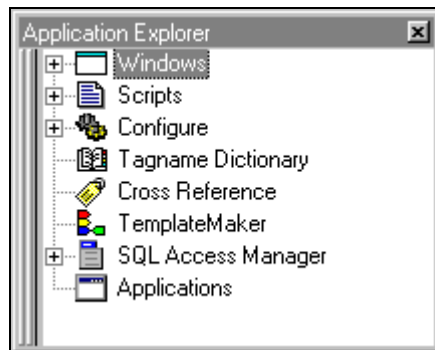
**Note** On the Windows NT operating system, you can configure the Application Explorer to launch any other program. This powerful feature allows you to quickly switch between your HMI configuration and I/O Server configuration.

Do not add WindowViewer (VIEW.EXE) to the Application Explorer. If you add WindowViewer, new windows you create in WindowMaker may not be synchronized with the windows in WindowViewer. The proper way to launch WindowViewer is by executing the **WindowViewer** command on the **File** menu, or by clicking the **Runtime** fast switch in the WindowMaker toolbar.

---

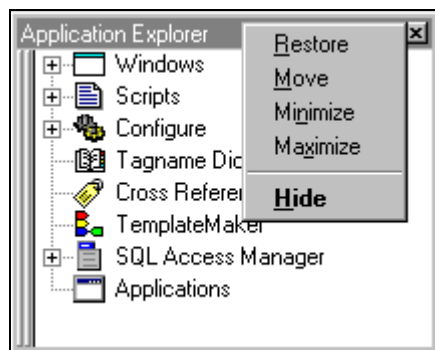
Like all WindowMaker's toolbars, the Application Explorer can be "docked" to any edge of the WindowMaker window or, "floated" anywhere within the WindowMaker window.

When you dock the Application Explorer to an edge of the WindowMaker window, it will automatically size itself accordingly and, if required, scroll bars will be displayed. When you float the Application Explorer within the WindowMaker window its title bar will be displayed. Like all WindowMaker toolbars, when the Application Explorer is floating, you can change its size. For example:



☞ For more information on docking/floating the Application Explorer see, "Working with the Floating/Docking Toolbars."

If you right-click the Application Explorer's title bar, the following menu appears:



☞ For more information on this menu see, "Right-Click Menus."

For more information on the right-click functionality within the Application Explorer, see "Navigating in the Application Explorer."



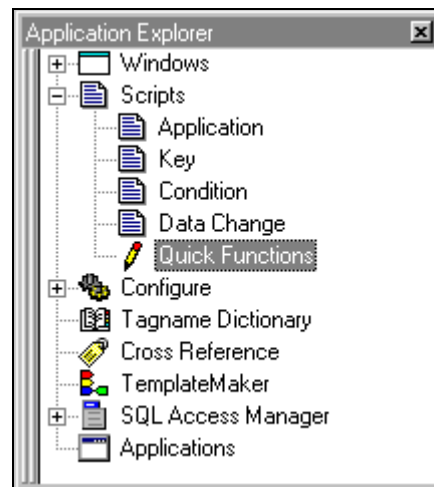
#### To show/hide the Application Explorer:

1. On the **View** menu, click **Application Explorer**. (When you initially start WindowMaker, by default, the Application Explorer is displayed.)
2. Repeat step 1 to close the Application Explorer.
  - ☞ To quickly hide the Application Explorer, click the Application Explorer tool on the **View** toolbar.
  - ☞ To quickly hide the Application Explorer when it is floating in the WindowMaker window, click the  button on its title bar or, right-click the title bar then, click **Hide** on the menu. When you show the Application Explorer again, it will reappear in its previous size and location in the window.

## Navigating in the Application Explorer


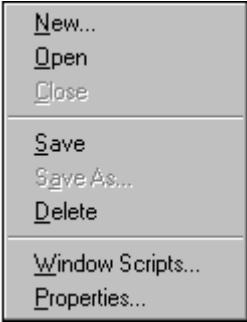



You can expand or collapse the groups listed in the Application Explorer hierarchical graphical view. For example, if you double-click on a group, the view will expand and display the group's members. If you double-click on a member, it will open that member. For example, in the **Windows** group, if you double-click on a member window name, the window will open. If you double-click on **Tagname Dictionary**, the **Tagname Dictionary** dialog box will appear, and so on.





- ☞ All groups that contain members will be preceded with a . You can click the  to quickly expand the group and view its members. Likewise, you can click the  to collapse the group and hide its members. For example:





The following section briefly describes the behavior of each group listed in the Application Explorer when you perform the described action:



Item	Action	Description
<b>Windows</b>	Double-click, or click 	Expands the view to display the names of all existing windows in your application. <ul style="list-style-type: none"> <li>• Double-click a window name to open it.</li> <li>• Right-click a window name to display a menu of commands that you can apply to the window. For example:</li> </ul> 
	Double-click, or click 	Collapses the group to hide its members.
<b>Scripts</b>	Right-click	Displays a <b>New</b> button. Click <b>New</b> to open the <b>Windows Properties</b> dialog box to create a new window.
	Double-click, or click 	Expands the view to display all OpenHMI QuickScript types: <ul style="list-style-type: none"> <li>Application</li> <li>Key</li> <li>Condition</li> <li>Data Change</li> </ul> <ul style="list-style-type: none"> <li>• Double-click a QuickScript type to open it.</li> <li>• Right-click a QuickScript type, an <b>Open</b> button appears. Click <b>Open</b> to open the QuickScript.</li> </ul>
	Double-click, or click 	Collapses the group to hide its members.


Item	Action	Description
<b>Configure</b>	Double-click, or click 	Expands the view to display many of WindowMaker's configuration commands and the <b>Wizard/ActiveX Installation</b> command. <ul style="list-style-type: none"> <li>• Double-click a configuration item to open its respective dialog box.</li> <li>• Right-click a configuration item, an <b>Open</b> button appears. Click <b>Open</b> to open the item's respective dialog box.</li> </ul>
<b>Tagname Dictionary</b>	Double-click, or click  Double-click	Collapses the group to hide its members. Opens the <b>Tagname Dictionary</b> dialog box displaying the last modified tagname's definition. Otherwise, the default <b>\$AccessLevel</b> system tagname is displayed.
<b>Cross Referencing</b>	Right-click Double-click Right-click	Displays an <b>Open</b> button. Click <b>Open</b> to open the <b>Tagname Dictionary</b> dialog box displaying the last modified tagname's definition. Otherwise, the default <b>\$AccessLevel</b> system tagname is displayed. Opens the <b>Cross Reference</b> utility. Displays an <b>Open</b> button. Click <b>Open</b> to open the <b>Cross Reference</b> utility.
<b>Add-on Programs</b>	Double-click, or click 	Expands the view to display the add-on program's configuration commands. Double-click a command to open its respective dialog box. <ul style="list-style-type: none"> <li>• Right-click a command, an <b>Open</b> button appears. Click <b>Open</b> to open the command's respective dialog box.</li> </ul>
	Double-click, or click 	<hr/> <b>Note</b> The add-on programs must be installed to appear in the Application Explorer. <hr/> Collapses the group to hide its members.

Item	Action	Description
<b>Applications</b>	Double-click, or click 	Expands the view to display all other applications that you can launch from WindowMaker. <ul style="list-style-type: none"> <li>• Double-click an application to launch it without exiting WindowMaker.</li> <li>• Right-click an application name to display a menu of commands that you can apply to the application.</li> </ul>
	Right-click	Displays a <b>New</b> button. Click <b>New</b> to add an application to the Application Explorer (see below).
	Double-click, or click 	Collapses the group to hide its members.

## Adding Applications to the Application Explorer

One of the most powerful features of the WindowMaker Application Explorer, on the Windows NT operating system, is its ability to launch other third-party Windows applications from within WindowMaker.

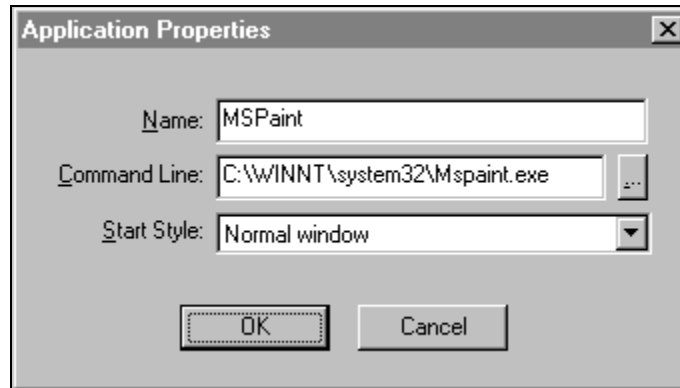
For example, you can run your I/O Server program and configure it at the same time that you are developing your application. You can launch third-party Windows programs that you frequently use such as Windows Notepad, Wordpad, Microsoft Excel, Microsoft Word, Microsoft Paint, and so on.

 The OpenHMI Recipe Manager add-on programs is automatically added to the Application Explorer when you install it.

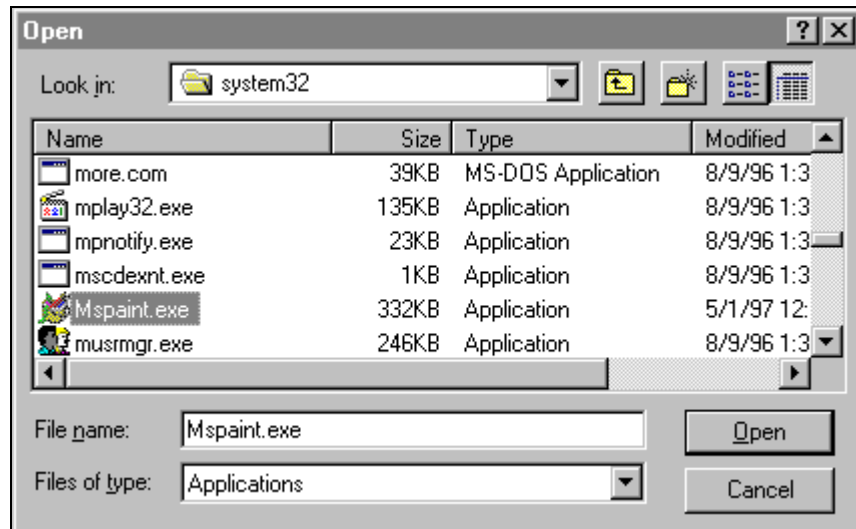
You can also configure the Application Explorer to open a specific document or spreadsheet in a program. For example, if you select a specific Microsoft Word document or Microsoft Excel spreadsheet, when you double-click that application's icon in the Application Explorer, the application will start up and automatically display the document or spreadsheet that you selected. These documents display the icon of the application in which they were originally created, or the .EXE configured as the associated application.

### ➤ To add an application to the Application Explorer:

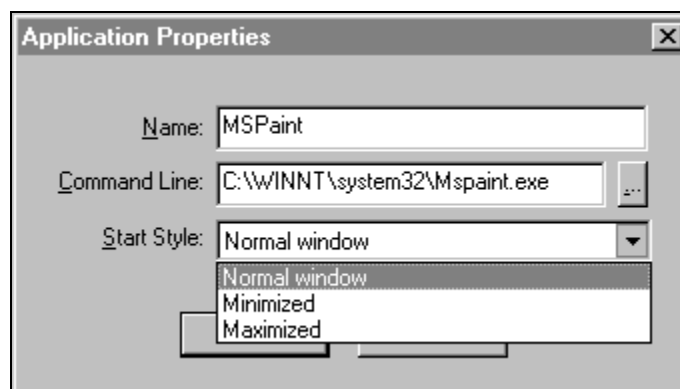
1. Display the Application Explorer.
2. Right-click **Applications**. A **New** button appears.
3. Click **New**. The **Application Properties** dialog box appears:



4. In the **Name** box, type the name that you want to display in the Application Explorer for the application.
5. In the **Command Line** box, type the full path for the application or, click the Ellipsis button. The **Open** dialog box appears:



6. Locate the application then, click **Open**. The **Application Properties** dialog box reappears:



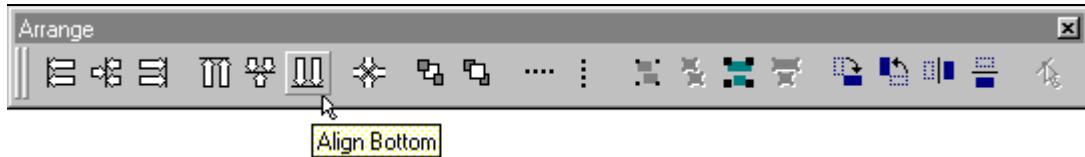
7. Click the **Start Style** arrow and select the style that you want for the application when you run it from WindowMaker.

8. Click **OK**. The application is added to the Application Explorer under **Applications**. You can now run the application at any time from WindowMaker.

## The WindowMaker Toolbars

The tools on the WindowMaker toolbars are grouped by common functionality. For example, the **Arrange** toolbar contains tools that you can use to quickly apply most of the commands found on the **Arrange** menu.

If you rest the cursor on a tool, a tool tips box will appear displaying the name of the tool. For example:



## Working with the Floating/Docking Toolbars

The WindowMaker toolbars have "floating and docking" capability. Meaning you can move any toolbar from its default "docked" position and dock it again on any edge of the WindowMaker window or, in the toolbar area at the top of WindowMaker's window. Docked toolbars can also be moved from their docked position at the edge of the window and floated within the window. When a toolbar is floating, it will have a title bar and you can change its size.

☞ The Application Explorer can also be docked or floated anywhere in the window and its size can also be changed when it is floating just like any other toolbar.

☞ For more information on the Application Explorer, see "The Application Explorer."

### ➤ **To change a docked toolbar's location in the window:**

1. Click the toolbar's "cool bars" or, on a blank area of the docked toolbar.
2. Hold down the left mouse button as you move the toolbar away from the edge of the window or, out of the toolbar area, or any edge of the WindowMaker window.
3. Move the toolbar to another edge of the window or, to a new position in the toolbar area.

☞ If you move a horizontally docked toolbar to the left or right edge of the WindowMaker window, it will automatically change to its default vertical shape when in position for docking to that edge. Likewise, if you move a vertical toolbar to the toolbar area at the top of the window or, to the bottom edge of the window, it will change to its default horizontal shape when in position for docking.


4. Release the mouse. The toolbar will be docked in the new location.

☞ When a toolbar is docked, you cannot change its size nor can you access its right-click menu.

### ➤ **To show/hide a docked toolbar:**

1. On the **View** menu select the toolbar's name. (When you initially start WindowMaker, by default, all toolbars are showing.)
2. Repeat step 1 to reverse your selection.

- ☞ When you show a docked toolbar that has been hidden again, it will reappear in its last docked location in the window.

➤  **To float a docked toolbar:**

1. Click the toolbar's "cool bars" or, a blank area of the docked toolbar.
2. Hold down the mouse button as you move the toolbar from its docked position to a new location within the WindowMaker window.
3. Release the mouse button. The toolbar will appear as follows:



- ☞ When a toolbar is floating, you can change its size.

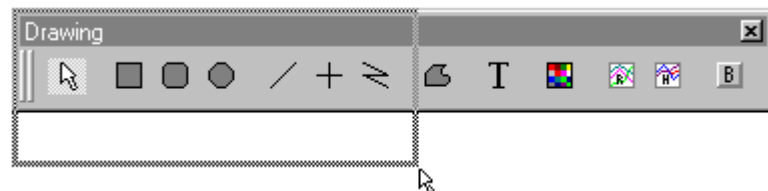
When you dock a floating toolbar on the left or right edge of the WindowMaker window, it will automatically change to its default vertical shape when it is in position for docking to that edge. Likewise, if you move it to the toolbar area at the top of the WindowMaker window or, to the bottom edge of the WindowMaker window, it will change to its default horizontal shape when in position for docking.

This will also occur when you dock a floating toolbar whose size you have changed. However, when you float the toolbar in the window again, it will return to its previous floated size.

➤ **To change the size of a floating toolbar:**

1. Move the mouse over any edge of the toolbar. The cursor will change to a double-ended arrow.
2. Click on the edge and hold down the mouse button as you move the mouse to size the toolbar.

- ☞ As you move the mouse, a box will appear to indicate the size the toolbar will be if you release the mouse button. For example:




3. Release the mouse button when the toolbar is the desired size.

- ☞ When you right-click the title bar of a floating toolbar, a menu appears displaying the commands you can apply to the toolbar.

☞ For more information on this menu, see "Right-Click Menus."


➤ **To hide/show a floating toolbar:**

1. To hide a floating toolbar, on the **View** menu, select the toolbar's name or, right-click the toolbar's title bar then, click **Hide** on the menu.

- ☞ To quickly hide the toolbar, click the  button on the toolbar's title bar.

2. To show a hidden floating toolbar, on the **View** menu select the toolbar's name.

- ☞ The toolbar will reappear at its previous location and in its previous size.

-  **To hide all toolbars at once:**
1. On the **View** menu, click **Hide All** or, click the Hide/Restore All tool on the **View** toolbar. All toolbars and the Application Explorer will be hidden.
  2. Repeat step 1 to show them again.
    - ☞ All displayed toolbars will have a check mark preceding their names on the **View** menu.

The following section briefly describes each WindowMaker toolbar and each tool it contains.









## General Toolbar

The **General** toolbar is grouped with tools that execute most of the window commands found on the **File** menu and the Microsoft Windows Clipboard tools found on the **Edit** menu:



- ☞ When you right-click a blank area of an open window, or right-click a window name under **Windows** in the Application Explorer, a menu appears that also contains most of these same windows commands.

☞ For more information on right-click menus, see "Right-Click Menus."

Button	Description
	Executes the <b>New Window</b> command on the <b>File</b> menu to open the <b>Windows Properties</b> dialog box to create a new window.
	Executes the <b>Open Window</b> command on the <b>File</b> menu to open the <b>Windows to Open</b> dialog box listing the names of existing windows that you can select to open.
	Executes the <b>Close Window</b> command on the <b>File</b> menu to open the <b>Windows to Close</b> dialog box listing the names of all currently open windows that you can select to close.
	Executes the <b>Save Window</b> command on the <b>File</b> menu to open the <b>Windows to Save</b> dialog box listing the names of all currently open windows that have been modified since they were last saved.
	Automatically saves all currently open windows that have been modified since they were last saved. This tool does not ask for confirmation on a per window basis. It saves all modified windows automatically.
	Executes the <b>Duplicate</b> command on the <b>Edit</b> menu to duplicate the currently selected object(s) in the window.
	Executes the <b>Cut</b> command on the <b>Edit</b> menu to cut the currently selected object(s) from the window and copies them to the Windows Clipboard.
	Executes the <b>Copy</b> command on the <b>Edit</b> menu to copy the currently selected object(s) and copies them to the Windows Clipboard. (Copied objects are not erased from the window.)





Executes the **Paste** command on the **Edit** menu to paste any object that has been cut or copied to the Windows Clipboard. (The cursor changes to the paste mode. Click in the window to paste the copied or cut object.)



Executes the **Undo** command on the **Edit** menu to reverse (undo) the last action or command applied to an object.



Executes the **Redo** command on the **Edit** menu to reverse (redo) last undo action or command applied to an object.

---

**Note** By default the number of undo/redo levels is set to 10. You can increase the **Levels of Undo** to 25 in the **WindowMaker Properties** dialog box. To access this dialog box, in the Application explorer, under **Configure**, double-click **WindowMaker** or, on the **Special** menu, point to **Configure** then, click **WindowMaker** on the



Executes the **Print** command on the **File** menu to open the **WindowMaker Printout** dialog box used to print database and window information and, QuickScripts.

## Wizards/ActiveX Toolbar

The **Wizards/ActiveX** toolbar, by default, only contains the wizard tool that you use to access the wizard **Selection Dialog** box. However, you can add any installed wizard or ActiveX control to the toolbar. For example:

















Displays the **Wizard Selection** dialog box used for selecting wizard to paste into your windows.

## Format Toolbar

The **Format** toolbar is grouped with tools that execute most of the text object formatting commands found on the **Text** menu. It also contains the tools that you will use to access the color palette to select line, text, fill, window background and transparent object color.









Button	Description
	Executes the <b>Font</b> command on the <b>Text</b> menu to open the <b>Font</b> dialog box used to select the font, its style and size.
	Executes the <b>Bold</b> command on the <b>Text</b> menu to apply <b>bold</b> styling to single or multiple text string selections and numeric value fields.
	Executes the <b>Italic</b> command on the <b>Text</b> menu to apply <i>italic</i> styling to single or multiple text string selections and numeric value fields.
	Executes the <b>Underline</b> command on the <b>Text</b> menu to apply <u>underline</u> styling to single or multiple text string selections and

	numeric value fields. Executes the <b>Reduce Font</b> command on the <b>Text</b> menu to reduce the point size of any font. This command can be applied by selecting the text string(s) and clicking on this tool on the <b>Text</b> toolbar.
	Executes the <b>Enlarge Font</b> command on the <b>Text</b> menu to enlarge point size of any font. This command can be applied by selecting the text string(s) and clicking on this tool on the <b>Text</b> toolbar.
	Executes the <b>Left Justified</b> command on the <b>Text</b> menu to align the left edge of single or multiple text string selections and numeric value fields.
	Executes the <b>Centered</b> command on the <b>Text</b> menu to center single or multiple text string selections and numeric value fields.
	Executes the <b>Right Justified</b> command on the <b>Text</b> menu to align the right edge of single or multiple text string selections and numeric value fields.
	Opens the color palette used to select the color for a line object or an object's outline.
	Opens the color palette used to select an object's fill color.
	Opens the color palette used to select the color for a text object.
	Opens the color palette to select a window's background color.
	Opens the color palette to select a transparent color for a bitmap object.

## Draw Object Toolbar

The **Draw Object** toolbar is grouped with all the tools that you will use to draw both simple graphic objects such as rectangles, ellipses, lines or text objects and, complex objects such as real-time trends, bitmaps and 3-dimensional buttons with labels in your windows:



Button	Description
	Selector mode used to select objects in the window.
	Rectangle tool used to draw rectangles or squares.
	Rounded rectangle tool used to draw rectangles or squares with rounded corners.
	Ellipse tool used to draw ellipses or circles.
	Line tool used to draw lines at any angle.
	Line tool used to draw horizontal or vertical lines.



Line tool used to draw polylines.



Polygon tool used to draw polygon objects.



Text tool used to type text objects.



Bitmap tool used to draw a bitmap container for pasting a bitmap directly from the Windows Clipboard or one of the following file types: .BMP, .JPG, .PCX or .TGA.



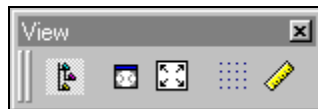
Real time trend tool used to draw real time trend objects.



Button tool used to draw a 3-dimensional button with a label.

## View Toolbar

The **View** toolbar is grouped with tools that execute most of the window commands found on the **View** menu. These commands are used to control the state of the WindowMaker window.



### Button

### Description



Turns on and off the **Application Explorer** command on the **View** menu to show/hide the Application Explorer.



Toggles the **Hide All** command on the **View** menu on and off to hide/show all docked toolbars.

When the hide all mode is active, the overall size of WindowMaker remains the same. To return to normal mode, click the Hide/Restore All tool on the floating **View Toolbar**, or click **Hide All** on the **View** menu.

☞ In the hide all mode, all floating toolbars remain visible and the **View Toolbar** automatically floats on top of WindowMaker. If you dock any of the floating toolbars while in the hide all mode, the mode is automatically terminated.



Toggles **Full Screen** command on the **View** menu on and off to switch the display mode from normal view to full screen.

To return to normal mode, click the Full Screen tool on the floating **View Toolbar**, or click **Full Screen** on the **View** menu.

☞ In the full screen mode, all WindowMaker program elements are hidden except, any open windows and floating toolbars. The **View Toolbar** automatically floats on top of WindowMaker.

In the full screen mode, the coordinates of the client area will remain the same. For example, the top left is 0,0. The full screen mode automatically sets the coordinates after it maximizes the client area, hides the Title Bar and menu bar and, adjusts the client area to mimic View's full screen mode.



Toggles the **Snap to Grid** command on the **Arrange** menu on and off to show/hide the visible grid used to align objects. Works with the **Snap to Grid** command on the **Arrange** menu.

☞ If the **Snap to Grid** option in the **WindowMaker Properties** dialog box is not selected, this tool will have no effect.



Turns on and off the **Ruler** command on the **View** menu to show/hide the ruler.










☞ For more information on the ruler, see "The WindowMaker Ruler."

## Arrange Toolbar

The **Arrange** toolbar is grouped with tools that execute most of the object arranging commands found on the **Arrange** menu:



The following briefly describes each tool:

Button	Description
	Executes the <b>Align Left</b> command on the <b>Arrange/Align</b> submenu. Aligns the left edge of all selected objects with the left edge of the left most selected object.
	Executes the <b>Align Center</b> command on the <b>Arrange/Align</b> submenu. Aligns the vertical centerline of all selected objects with the centerline of the group of objects selected.
	Executes the <b>Align Right</b> command on the <b>Arrange/Align</b> submenu. Aligns the right edge of all selected objects with the right edge of the right most selected object.
	Executes the <b>Align Top</b> command on the <b>Arrange/Align</b> submenu. Aligns the top edge of all selected objects with the top edge of the top most selected object.
	Executes the <b>Align Middle</b> command on the <b>Arrange/Align</b> submenu. Aligns the middle of all selected objects with the middle of the group of objects.
	Executes the <b>Align Bottom</b> command on the <b>Arrange/Align</b> submenu. Aligns the bottom edge of all selected objects with the bottom edge of the lowest selected object.
	Executes the <b>Align Centerpoints</b> command on the <b>Arrange/Align</b> submenu. Aligns the centerpoint of all the selected objects with the centerpoint of the group of selected objects.
	Executes the <b>Send to Back</b> command on the <b>Arrange</b> menu to place all selected objects behind all objects that are not selected.
	Executes the <b>Bring to Front</b> command on the <b>Arrange</b> menu to place all selected objects in front of all objects that are not selected.



Executes the **Space Horizontal** command on the **Arrange** menu to evenly space all selected objects horizontally between the left most and right most selected objects.



Executes the **Space Vertical** command on the **Arrange** menu to evenly space all selected objects vertically between the top most and bottom most selected objects.



Executes the **Make Symbol** command on the **Arrange** menu to combine multiple objects into a single unit called a symbol..



Executes the **Break Symbol** command on the **Arrange** menu to break a symbol into its individual components..



Executes the **Make Cell** command on the **Arrange** menu to combine multiple selected objects into a single unit called a cell. When combining cells each cell will be retained, so when the combined cell is broken the original cells are restored..



Executes the **Break Cell** command on the **Arrange** menu to break a selected cell. When combining cells each cell will be retained, so when the combined cell is broken the original cells are restored..



Executes the **Rotate Clockwise** command on the **Arrange** menu to rotate selected objects clockwise 90 degrees:.



Executes the **Rotate CounterClockwise** command on the **Arrange** menu to rotate selected objects counter clockwise 90 degree.



Executes the **Flip Horizontal** command on the **Arrange** menu to flip selected objects horizontally.



Executes the **Flip Vertical** command on the **Arrange** menu to flip selected objects vertically.

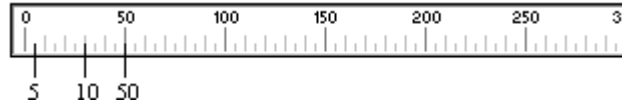


Turns on the **Reshape Object** command on the **Edit** menu to reshape a polygon or polyline.

## The WindowMaker Ruler

You can use the WindowMaker ruler to do precision alignment of the objects in your windows.

The small tick marks are spaced 5 pixels apart. The medium tick marks are spaced 10 pixels apart. The numbered large tick marks are spaced 50 pixels apart. For example:



The ruler's 10 and 50 pixel spacing increments are equivalent to the distance in pixels that a selected object is moved when you hold down the SHIFT or CTRL key and press an up, down, right or left arrow key.

For example, if you want to move an object 10 pixels at a time, hold down the SHIFT key while you press an arrow key. To move an object 50 pixels at a time, hold down CTRL key while you press an arrow key.

☞ When you select an object and only press an arrow key, the object is moved 1 pixel at a time.

These features can be useful when you need to make fine alignment and location adjustments.

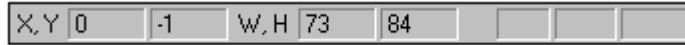
☞ For more information on the using the arrow keys, see "Moving Objects with the Arrow Keys."

➤  **To show/hide the ruler:**

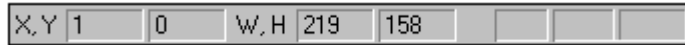
1. On the **View** menu, click **Ruler** or, click the ruler tool in the **View** menu.
2. Repeat step one to hide the ruler.

## The WindowMaker Status Bar

When you select an object in a window, the WindowMaker status bar displays the object's upper left X and Y pixel coordinates and the object's pixel height and width. For example:



When you select multiple objects, the status bar displays the coordinate for the left edge of the left most object (X) and the coordinate for the top edge of the top most object (Y). The width and height are also show for the entire group. For example:



When you click a blank area of a window the status bar displays the X and Y coordinates for the current location of the cursor in the window. For example:



- **To show/hide the status bar:**
1. On the **View** menu, click **Status Bar**.
  2. Repeat step one to hide the **Status Bar**.

## The WindowMaker Color Palette

The WindowMaker palette that provides 16.7 million color support. (The color support is limited only by your video card capability.) By default, the palette offers you a wide range of color selections. However, you can create your own custom palettes. Your custom palettes can be loaded into and exported from the WindowMaker color palette.

### Using the Standard Color Palette

The WindowMaker color palette is used to apply color to static and dynamic properties of lines, rectangles, round rectangles, ellipses, polylines, polygons and text. It is also used to select the background color for your windows and the transparent color for bitmaps that allows you to view objects behind bitmaps.

For more information on transparent bitmaps, see Chapter 2 - Using WindowMaker.

The color palette appears whenever you click a colored square in a dialog box or, you click one of the color tools to apply line, fill or text color to a selected object.

➤ **To use the standard color palette:**

1. To select a standard color, click the color that you want to use in the **Standard Palette** section. (The color palette will close and the color you selected will be applied.)



2. To select one of OpenHMI's 32 classic colors (palette colors prior to OpenHMI Version 7.0), click the >> in the right corner. The **Classic Colors** will appear:



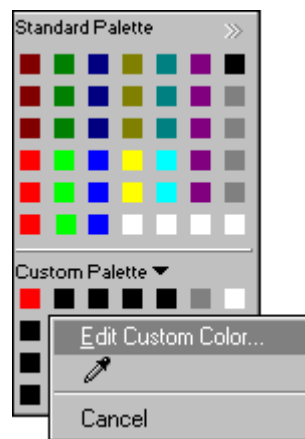


## Creating a Custom Color Palette

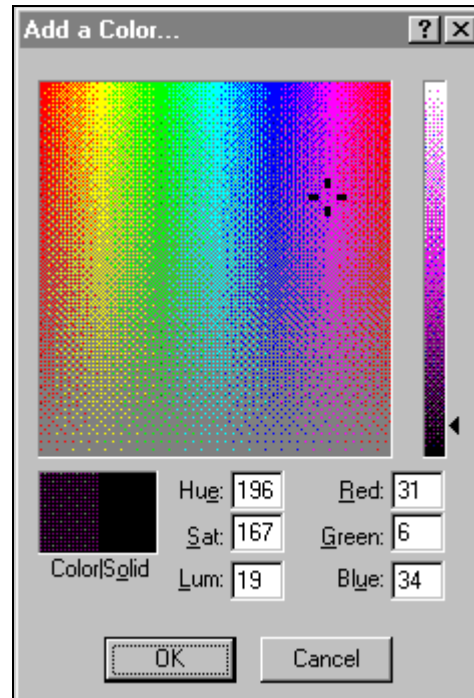
The WindowMaker color palette allows you to define custom colors and add them to your palette. It also allows you to import palettes created in other windows applications and add them to the standard palette. You can also export your custom palettes to other windows applications.

➤ **To create a custom color:**

1. Open the color palette.
2. Right-click one of the blank squares in the **Custom Palette** section at the bottom of the color palette. The following menu appears:



3. Click **Edit Custom Color**. The **Add a Color** dialog box appears:



- Click anywhere in the matrix then, use the slider at the right of the dialog box to adjust the color's attributes.

#### **Hue, Sat, Lum**

A combination of hue, saturation and luminosity can be used to define any color.

Hue is the value of a color wheel, where 0 is red, 60 is yellow, 120 is green, 180 is cyan, 200 is magenta and 240 is blue. Saturation is the amount of color in a specified hue, up to a maximum of 240. Luminosity is the brightness of a color. If you change any of these values, the red, green and blue scales will change to match.

The easiest way to experiment with different colors is to press and hold the mouse then, move the cursor around in the color matrix.

#### **Red, Green, Blue**

A combination of red, green and blue levels can be used to define any color. You can see the effect of changing these values in the color matrix. If you change these values, the values for hue, saturation and luminosity will change to match.

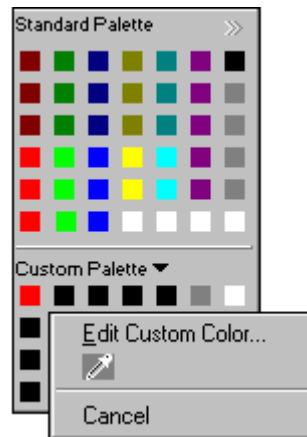
If you define a color using the **Hue, Sat, Lum** or **Red, Green, Blue** scales, you can view the color in the **Color|Solid** boxes to make sure you defined the color as you intended.

The **Color** (left) box shows the amount of white and black in the color you specified. The **Solid** (right) box, shows how the color will look if you choose 100% of the color with no white and black. To adjust the color, use the slider at the right of the dialog box. To specify that you want 100% of the color with no white or black, type ALT + O.

- Click **OK**. The color you selected will be added to the square that you originally clicked in the color palette.

#### ➤ **To select a custom color with the blotter tool:**

1. Open the color palette.
2. Right-click one of the blank squares in the **Custom Palette** section at the bottom of the color palette. The following menu appears:

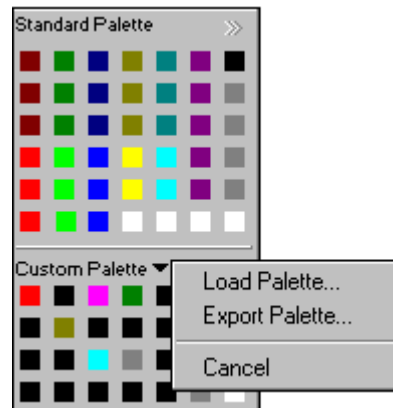


3. Click the blotter tool then, click the color that you want to add to the **Custom Palette** section of the color palette. You can select any color anywhere within the WindowMaker window or outside of WindowMaker completely. This feature is primarily intended to be used when you are creating transparent bitmaps.

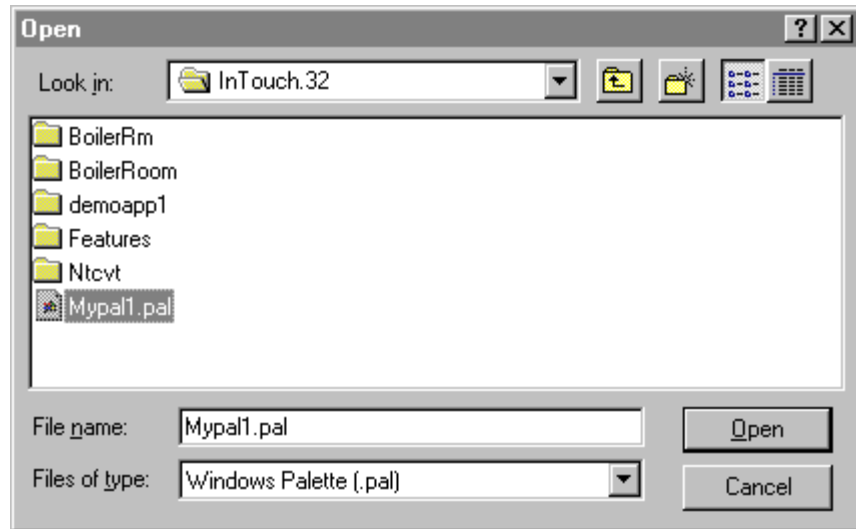
*For more information on creating transparent bitmaps, see Chapter 2 - Using WindowMaker.*

➤ **To import a custom palette:**

1. Open the color palette.
2. Click the **Custom Palette** arrow. The following menu appears:



3. Click **Load Palette**. The standard Windows **Open** dialog box appears: .



4. Locate and select the palette (.PAL) file then, click **Open** or double-click the file name. The colors contained in your palette will be loaded into the **Custom Palette** section of the color palette.
- **To export a custom palette:**
1. Open the color palette.
  2. Click the **Custom Palette** arrow then, click **Export Palette** on the menu (shown above).
  3. The standard Windows **Save As** dialog box will appear. Specify the name that you want to save the palette as then, click **Save**.
    - ☞ The palette must be saved with the .PAL extension.

## Right-Click Menus

OpenHMI supports right-click mouse functionality to display menus for frequently used commands for windows and graphic objects. Right-click menus containing the commands that can be applied to selected text in dialog box, text box, an animation link tagname or expression box or, in a QuickScript window is also supported. Instead of using the standard menus to find the command you want to use, you can simply right-click the window, object, dialog box text box, or the groups and their members in the Application Explorer.

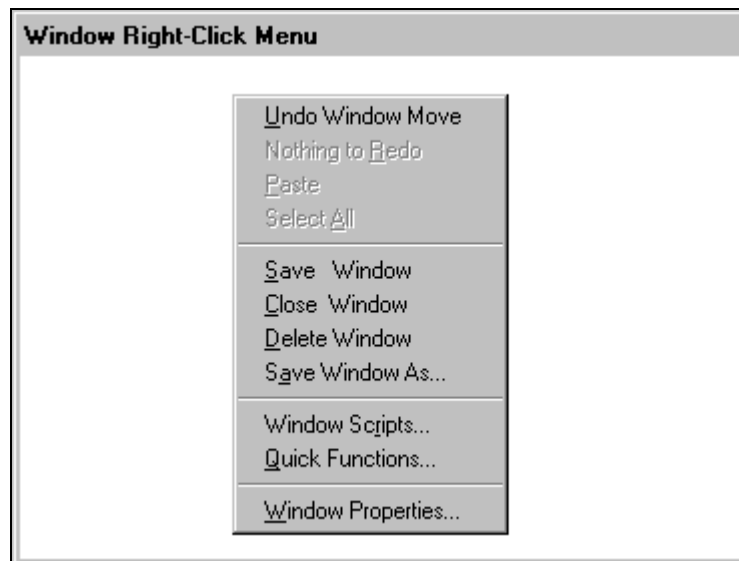
---

**Note** To turn off the right-click functionality, add the line **oldrightmousebehavior=1** to your OPENHMI.INI file.

---

➤ **To access the window right-click menu:**

1. Right-click a blank area of a window. The following menu appears:



2. Click the command that you want to use on the menu.

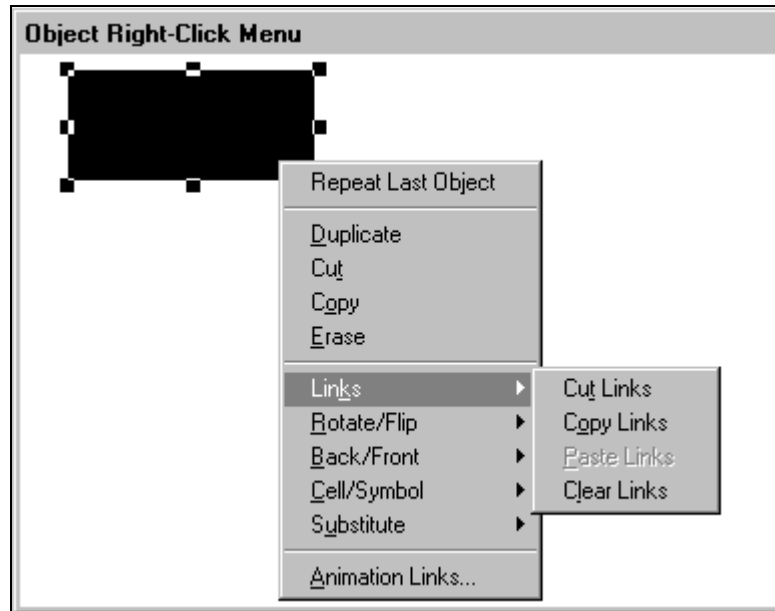
---

**Note** Commands that are not applicable for the current state of the window will not be active.

---

➤ **To access the graphic object right-click menu:**

1. Right-click an object in the window. The following menu appears:



2. Click the command that you want to use on the menu.

☞ If a "submenu" exists for the command, an arrow will be displayed. To select a command in a submenu, point at the command in the initial right-click menu then, click the command that you want to apply to the object in the submenu.

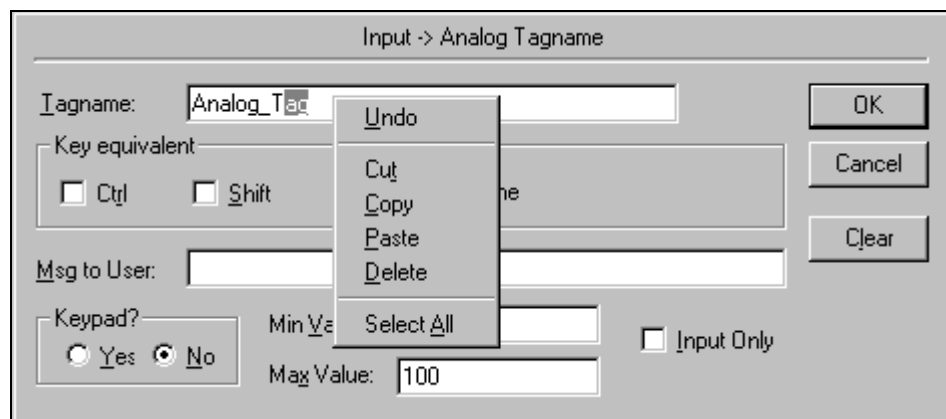
---

**Note** Commands that are not applicable for the current state of the window will not be active.

---

➤ **To access the dialog box text right-click menu:**

1. In any WindowMaker dialog box, right-click a text box. The following menu appears:



2. Click the command that you want to apply to the selected text.

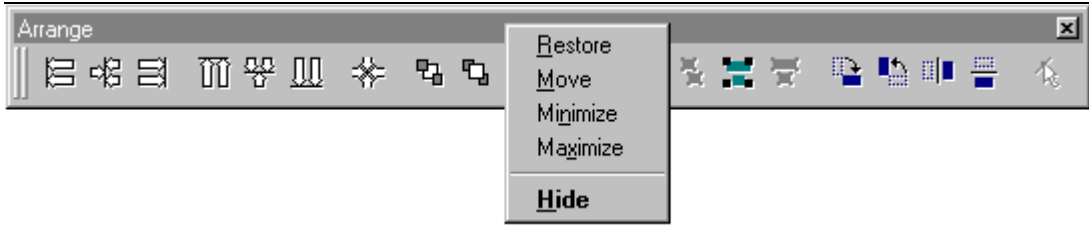
---

**Note** Commands that are not applicable for the current state of the text will not be active. The **Select All** command will become active when partial or no text is selected.

---

➤ **To access a floating toolbar right-click menu:**

1. Right-click the floating toolbar's title bar. The following menu appears:

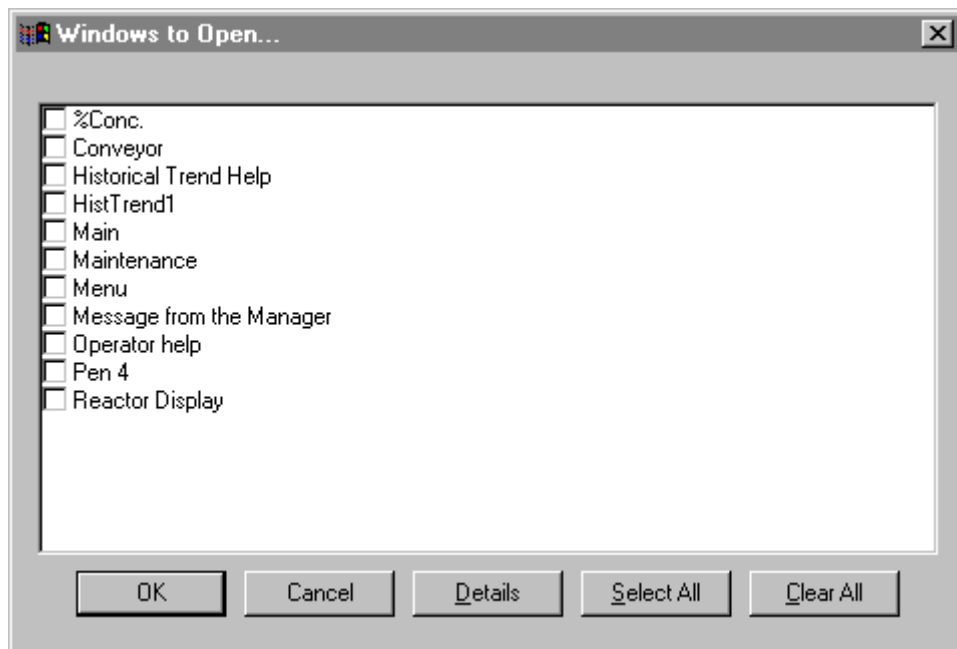


2. Click the command on the menu that you want to apply to the toolbar.

## Common Window Dialog Box Features

When you are opening, saving, closing, deleting or duplicating a window(s) using the commands on the **File** menu in WindowMaker, the dialog boxes that you will use are very similar and have many common features. To avoid redundancy in the procedures describing how you perform these actions, the common features of those dialog boxes are described in this section.

When you right-click on a blank area of an open window, or you click the open, save, close, delete or save as window commands on the **File** menu, by default, the respective dialog box for the command you selected will appear in the "list view." Meaning that the names of all the windows that are applicable for the selected command will appear in a continuous list. For example:



---

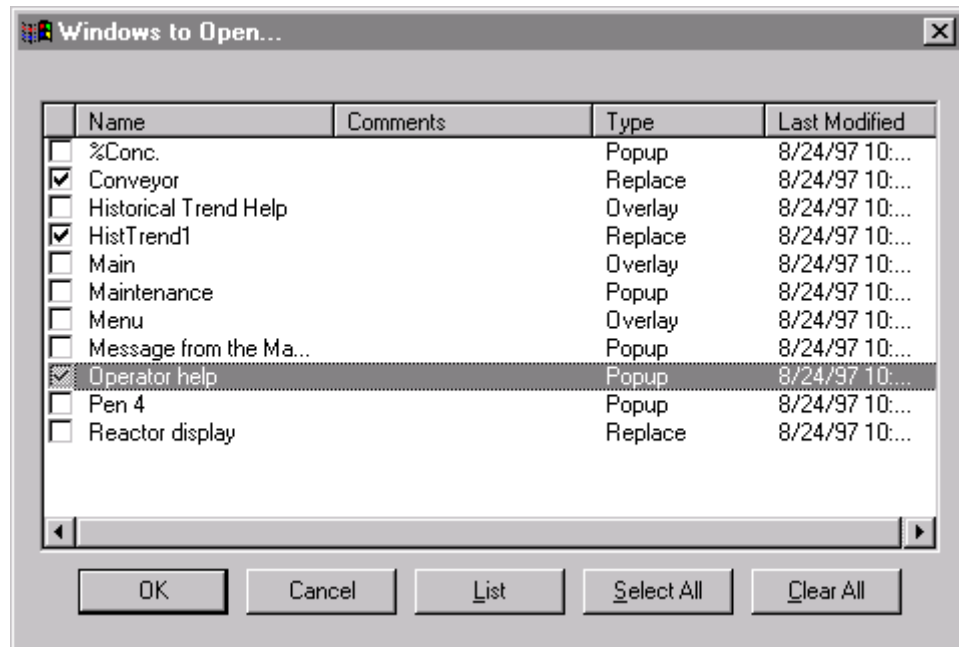
**Note** A horizontal scroll bar will appear when the number of window names exceeds the default list space.

---

Click **Details** to change from the "list view" to the details view.

When you select the details view, the windows and their details are displayed in a multi-column format. The details displayed include any comments you entered for the window in the **Window Properties** dialog box, the window's type, the date and time it was last modified. For example:





**Note** In the details view, you can select any unopened window by clicking on any portion of its row, not just the check box. (The entire row will be highlighted.) You can click on a selected window a second time, to deselect it.)

A vertical scroll bar will also appear when the number of window names exceeds the default list space.

To sort the list by a detail type, click the column header for that detail. The details view sort sequences:

- **Name** - Alphabetically
- **Comments** - Alphabetically
- **Type** - Overlay, Replace then Popup
- **Last Modified** - From oldest date/time (top) to most recent (bottom)

☞ Each time you click a column header, the list sort order will toggle from ascending to descending. For example, if the list is currently sorting in ascending order and you click a column header, the list will be resorted in descending order for the column selected.

To return the list to the default display, click the small box on the far left side of the column header.

To size the columns, place the cursor over the vertical lines that separate each detail header. When the cursor changes to an "I" bar, click and drag the header to the width you want for the column.

☞ To quickly auto-size a column, double-click on the column's right vertical line separator.

To open selected window(s) click **OK**.

To cancel your selections and close the dialog box, click **Cancel**.

To return the dialog box to "list view," click **List**.

To select all listed windows, click **Select All**.

To clear all selected windows, click **Clear All**.

## Miscellaneous Mouse Short Cuts

Double-clicking on any object or symbol automatically executes the **Animation Links** command (on the **Special** menu) with the object or symbol selected.

*ℳ* For more information on animation links, see Chapter 4 - Creating Animation Links.

Double-clicking in a blank expression input field within a link definition dialog box launches the Tag Browser listing all tagnames defined in the application's Tagname Dictionary.

Double-clicking after a period (.) in an expression input field on a link definition dialog box will display the **Choose field name** dialog box containing a global listing of all the tagname **.fields**.

Double-clicking on a tagname in an animation link tagname or expression opens that tagname's definition in the Tagname Dictionary.

*ℳ* For more information on the Tag Browser and tagname **.fields**, see Chapter 3 - Tagname Dictionary.

Right-clicking on a blank area of an open window, a text box in any WindowMaker dialog box or, on a graphic object will display a menu with the commands that you can apply to the window, text or object. Right-clicking the title bar of a floating toolbar will also display a menu of commands that you can apply to the toolbar.

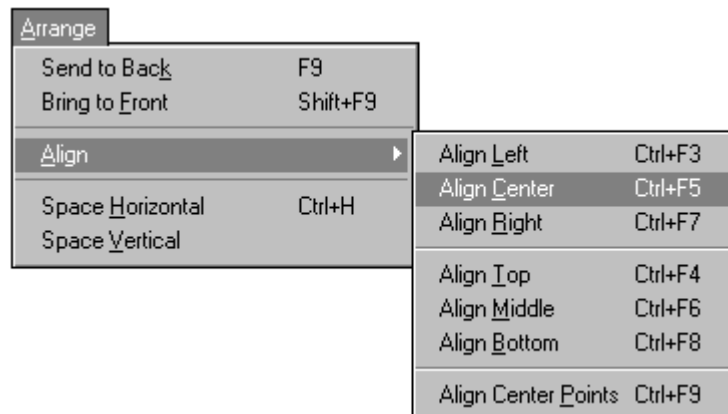
## Short Cuts and Accelerators

OpenHMI provides many mouse and keyboard shortcuts for frequently used functions. Whenever a menu command has a keyboard shortcut, it is displayed on the menu to the right of the command. In addition, all commands may be executed with a three-key sequence beginning with the ALT key. The second key is the underlined character in the menu name, and the third key is the underlined character in the command.

For example, you can execute the **N**ew **W**indow command on the **F**ile menu by using the sequence ALT + FN.

To execute a command on a right-click menu, type the underlined character in the command.

To execute a command on a submenu, you must press a three key sequence. For example, to execute the **A**lign **C**enter command on the **A**lign submenu you would press ALT + AAC, or CTRL+F5



---

**Note** When you select a menu command that is followed by an ellipsis (...), a dialog box appears, and you must select or enter more information in order to complete the command.

---

## Moving Objects with the Arrow Keys

In WindowMaker, you can use the up, down, left and right arrow keys to move a selected object or group of objects one pixel at a time in the direction of the arrow key being pressed. This feature can be very convenient when you need to make fine alignment and location adjustments. To move the object 10 pixels at a time, hold down the SHIFT key while pressing the arrow key. To move the object 50 pixels at a time, hold down the CTRL key while pressing the arrow key.

In WindowViewer, when you use the arrow keys, an algorithm is used to move from one touch-sensitive object to another. For example, the left cursor arrow key cuts a path to the left as wide as the height of the selected object. If it intersects any part of another touch-sensitive object within its path, that object takes the focus as the currently selected object. If no other touch sensitive object is encountered, the path wraps around to the left edge of the screen and continues to try to intersect another object. If no other object is encountered, the original object remains selected.

All arrow keys function in the same manner. The up and down arrow keys use the width of the selected object for the width of their paths.

☞ If you know that the user of your application will only use cursor keys to navigate in the windows in your application, you should carefully plan the placement of the touch-sensitive objects in the window to ensure that they have intersecting paths.

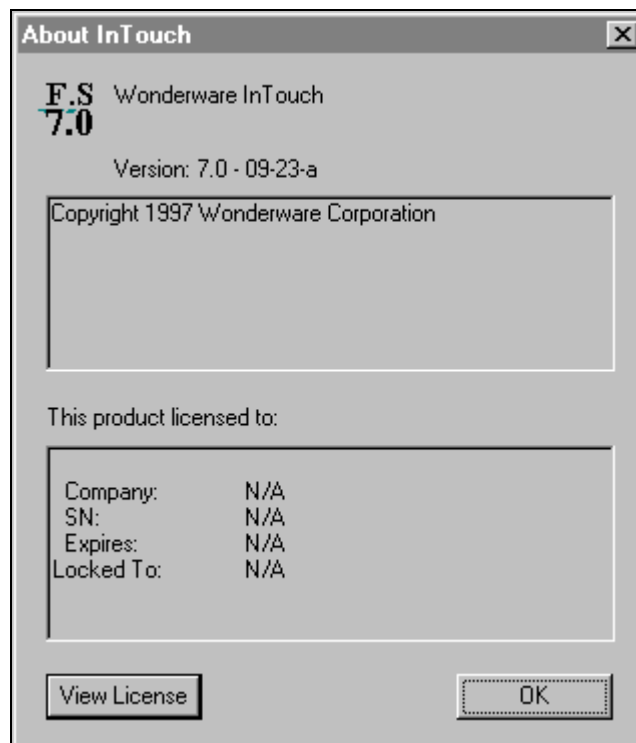
In WindowViewer, the TAB key can also be used to transition between the touch-sensitive objects in a window. (However, the tabbing order cannot be guaranteed.)

By careful application design, and understanding how the arrow keys react, you can create applications that can be used without a mouse.

## Using WindowMaker Help

WindowMaker supports context-sensitive help as follows:

- For help on an open dialog box, press the F1 key. The respective Windows online help topic will appear.
- For help on the various QuickScript functions, in the QuickScript editor, click **Help** or, on the **Insert** menu, point to **Functions** then, click **Help**. A dialog box will appear listing all available QuickScript functions. Click the function that you want help for. The function's respective Windows online help topic will appear.
- To obtain information about your OpenHMI software, for example, the Version you are using, your Serial Number and your License expiration date and so on, on the WindowMaker **Help** menu, click **About**. The **About OpenHMI** dialog box appears:



Click **View License** to launch the license viewing utility to obtain information regarding your OpenHMI license.

## CHAPTER 2

# Using WindowMaker

By setting various properties for WindowMaker and WindowViewer, you can customize the functionality and final appearance of your application. For example, you can specify what menus you want available in WindowViewer, you can include company names in the title bar of your application, and so on.

This chapter describes how to configure WindowMaker and WindowViewer, work with WindowMaker's windows, edit and arrange graphic objects, and how to install and use wizards and ActiveX controls.

## Simple Objects

WindowMaker has four basic types of simple objects; lines, filled shapes, text and buttons. Each of these simple object types have attributes that affect their appearance. These attributes include line color, fill color, height, width, orientation, etc. and can be either static or dynamic. A static attribute remains unchanged during the operation of the application. A dynamic attribute is linked to the value of an expression such that a change in the value of the expression results in a change in the attribute. For example, the fill color of an object can be linked to the value of a discrete expression. Based on the state of the expression, the fill would be one color when the expression is true and another color when it is false. Most of the attributes of simple objects can be made dynamic. An object may have more than one dynamic attribute. Dynamic attributes may be combined freely to achieve the desired result. The following briefly describes WindowMaker's simple object types:

Object	Description
<b>Line</b>	A line object is made up of one or more line segments depending on the type of line. Color is the only attribute of a line that you can animate. Width and style cannot be animated. They are simply assigned default attributes.
<b>Filled Shapes</b>	Filled shapes are two dimensional objects made up of a closed interior area surrounded by a line. Examples of filled shapes are rectangles, rounded rectangles, circles, ellipses and polygons. The attributes of a filled shape are: line color, line width, line style, fill color, percent color fill, height, width, location, visibility, orientation and size.
<b>Text</b>	Text is an object made up of a string of characters on a single line. The attributes of a text object are: font, size, color, <b>bold</b> , <u>underline</u> , <i>italic</i> , justification, visibility and location.
<b>Buttons</b>	The 3-dimensional buttons can be created for any desired size by using the Button tool on the WindowMaker <b>Draw Object Toolbar</b> . The default "text" string on the button face is edited by using the <b>Substitute Strings</b> command on the <b>Special</b> menu.  Many types of links can be attached to buttons such as action scripts, key scripts, analog or discrete value input or output links. If an input or output link is attached to a button the value is displayed on the button as the text string.

## Complex Objects

In addition to simple objects, OpenHMI also supports complex objects which are considerably different. The following briefly describes WindowMaker's complex object types:

Object	Description
<b>Bitmap</b>	The Bitmap tool is used to copy and paste bitmaps into your application. Once pasted in an application window, a bitmap can be rotated and, it can be defined with a transparent background so that it can float over other objects.
<b>Trends</b>	Display for real-time data that can be configured to display graphical representations of multiple tagnames over time.
<b>Symbols</b>	<p>A symbol is a combination of simple objects (lines, filled shapes, and text) which is treated as a single object. Any attribute change applied to a symbol, whether it is a change in a static attribute in WindowMaker, or a change in a dynamic attribute in WindowViewer, will affect all the component objects of a symbol.</p> <p>For example, if you create a pump symbol from two circles and two rectangles, and then you define a fill Color Link on the symbol, the fill color will apply to all four objects. Similarly, in WindowMaker, a Fill Color selection would also change the fill color of all the component objects.</p> <p>The component objects of a symbol can have different values for the same attribute if these attributes were different before the objects were combined into the symbol and were not changed after they became a symbol. Symbols cannot contain bitmaps, buttons, cells, alarms or trends.</p>
<b>Cells</b>	<p>A cell is a collection of two or more objects, symbols, or other cells that are joined together to form a single unit. Cells maintain a fixed spatial relationship between their individual graphic elements. Each component of a cell can have its own links. Cells are used to create virtual devices such as a slide controller.</p> <p>A cell is created by selecting two or more objects, symbols, and/or cells, and then executing the <b>Make Cell</b> command on the <b>Arrange</b> menu.</p> <p>Once a set of objects is made into a cell, its interior details such as colors, sizes, animation links, and so forth, cannot be changed. The only way to change a cell's appearance or operation is by "breaking" it apart using the <b>Break Cell</b> command on the <b>Arrange</b> menu.</p> <p>The attributes of the components of a cell are changed in WindowViewer by the operation of links. Cells can be duplicated, copied, pasted, aligned, spaced, and so on. Cells cannot be sized. They must be broken apart, sized and made back into a cell. Cells are very useful in creating multiple similar devices that are associated with different tagnames.</p>
<b>Wizards</b>	Wizards save you a considerable amount of time during application development. They are easy to use and easy to configure. When you select a wizard and paste it into a



window, and then double-click it, its configuration dialog box appears and you can configure it.


For example, in the case of a "slider" wizard, the configuration would include items such as the tagname to effect, the minimum and maximum range labels for the slider, the fill color, and so on. Once the required configuration information is entered, the Wizard is ready to use. By using Wizards, you do not spend time drawing the individual components for the object, or entering the value ranges for the object, or animating the object.

"Complex" Wizards can be developed to provide "behind the scenes" types of operations. These operations may include creating complete display windows, creating or converting a database, importing an AutoCad drawing, and configuring other applications such as, the OpenHMI Recipe Manager.

---

**Note** A proficient "C" programmer can develop custom Wizards by using Xycom Automation's Extensibility Toolkit.

---

 For more information on Wizards, see "[Working with Wizards](#)."

### ActiveX Controls

WindowMaker supports ActiveX controls which, in their simplest form, are mini-applications that talk to or run within your application. WindowMaker supports all ActiveX controls that are included in the OpenHMI configurator software. WindowMaker also supports third-party ActiveX controls such as those installed with Office97.

You install the OpenHMI ActiveX controls just like any other wizard. If desired, you can add frequently used ActiveX controls to your WindowMaker **Wizards/ActiveX Toolbar**.

When you select an ActiveX control and paste it into a window, and then double-click it, its configuration dialog box appears. When you configure an ActiveX control, you specify a unique control name for it in order to reference it in a script (a default name is created when you initially add the control).

All ActiveX controls have properties, methods and events associated with them. You can associate an ActiveX property with a tagname of a corresponding data type. You can execute ActiveX methods through OpenHMI QuickScript functions. You can associate an ActiveX event with an ActiveX Event Script that executes when the event occurs. In other words, you can use OpenHMI QuickScript functions to handle control events, call control methods, and control properties.

In runtime, the tagnames and QuickScripts you defined in WindowMaker control the behavior of your ActiveX controls.

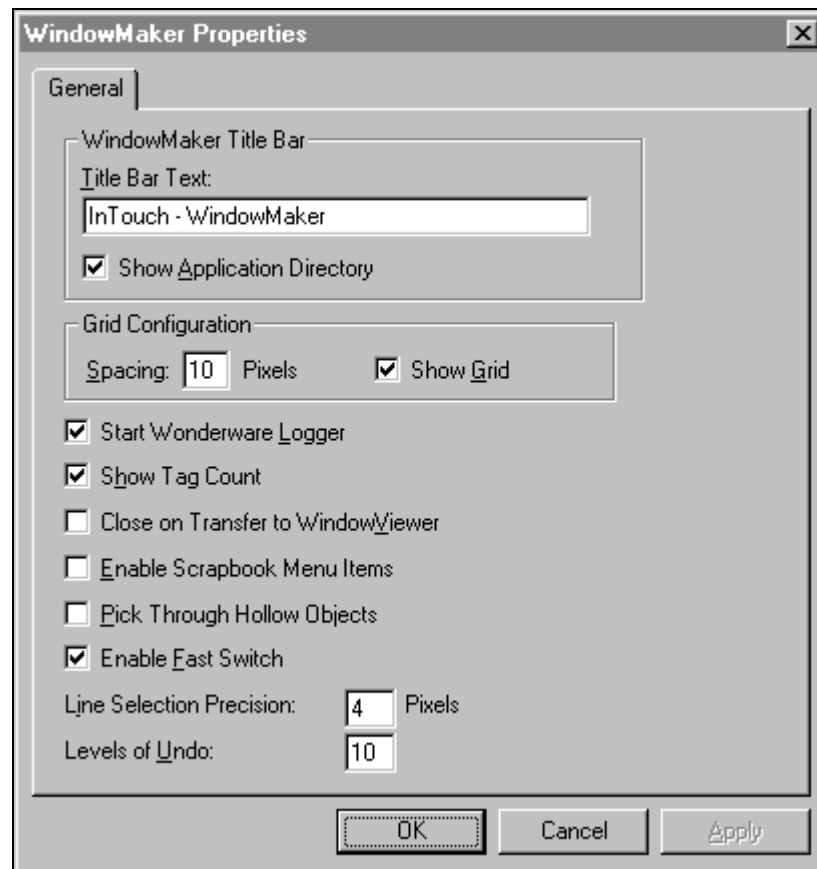
# Customizing Your Development Environment

There are many properties that you can set to customize WindowMaker. For example, you can customize your application's title bar text to include the company name. You can set the pixel spacing for the grid and so on.

➤ **To set the properties for WindowMaker:**

1. On the **Special** menu, point to **Configure**, and then click **WindowMaker**, or in the Application Explorer under **Configure**, double-click **WindowMaker**. The **WindowMaker Properties** dialog box appears with the **General** properties sheet active:

☞ In the Application Explorer under **Configure**, you can also right-click **WindowMaker**, and then click **Open**.



☞ If you right-click any of the text entry boxes in any dialog box, a menu will appear displaying the commands that you can apply to the selected text.

2. In the **Title Bar Text** box, type the title that you want to appear in your application's title bar in runtime. For example:

**ABC Company, Paint APP1**

---

**Note** You cannot change the title bar if you are using a "Promotional License."

3. **Show Application Directory** if you want to include the path to the application's directory in the title bar. For example:

**ABC Company, Paint APP1 - C:\DEMOAPP1**

4. In the **Spacing** box, type the number of pixels that you want spaced between the snap to grid's coordinates.
5. Select **Show Grid** if you want a visible grid in your windows when you turn on WindowMaker's "snap to grid" functionality.

☞ If you do not select **Show Grid**, no grid will be visible in your windows when you turn snap to grid on.



Click the Snap to Grid tool on the **View** toolbar or, on the **Arrange** menu, click **Snap to Grid** to turn the snap to grid functionality on and off.

Objects are snapped to the grid by their upper left corner. If you select multiple objects, the snapping will be applied to the upper left corner of the first object selected in the group.

6. Select **Start Wonderware Logger**, if you want the Logger application to automatically start when you start WindowMaker.

☞ The Logger's behavior is a little different when you are on the Windows NT Operating System. For more information on the Logger, see the "[Welcome to OpenHMI](#)" section.

7. Select **Show Tag Count**, if you want the number of tagnames defined in your Tagname Dictionary to be displayed in the WindowMaker menu bar.

☞ This can be very useful information if you are creating an application with a limited Tagname Dictionary size.

---

**Note** Showing the tagname count will seriously impact performance of the Tagname Dictionary in WindowMaker.

---

8. Select **Close on Transfer to WindowViewer**, if you want WindowMaker to automatically close when you start WindowViewer.

☞ If memory is not an issue and you are moving often between WindowViewer and WindowMaker, this option should not be selected.

---

**Note** When you select this option, the **Close WindowViewer** option located on the **WindowViewer Properties General Properties** sheet is automatically selected too.

---

9. Select **Enable Scrapbook Menu Items** if you are copying and pasting objects between WindowMaker and Scrapbook+.

---

**Note** The Windows Scrapbook+ is not supported by OpenHMI.

---

10. Select **Pick Through Hollow Objects**, if you want to be able to select objects that are behind "hollow" objects.

☞ If you select this option, and then draw four lines and join them to make a frame around another object, you can select the object within the frame without having to send the frame to the back.

11. Select **Enable Fast Switch**, if you want the fast switch between WindowMaker and WindowViewer to be displayed in your menu bar for both programs.

☞ If you select this option, in WindowMaker, the fast switch is the word **Runtime** displayed in the upper right hand of your screen. In

WindowViewer, it is the word **Development**. To quickly switch between the two programs, click the fast switch.

---

**Note** When you use the fast switch, WindowMaker automatically saves all changes made to all open windows before WindowViewer is started.

---

12. In the **Line Selection Precision** box, type the number of pixels that your cursor can be away from a line and still be able to select it.

☞ In most cases, the default setting of 4 should be sufficient.

13. In the **Levels of Undo** box, type the number of undo/redo levels that you want saved. You can have up to 25 levels. If you type zero, the undo/redo functionality is turned off.

One level represents one action. The undo and redo stacks are empty when you create a new window or open an existing window. Both stacks are emptied when you save the window.

☞ For more information on undo and redo, see "[Undoing Object Edits](#)."

14. Click **OK**.

---

**Note** After you modify any of these settings, you must restart WindowMaker to apply your changes.

---

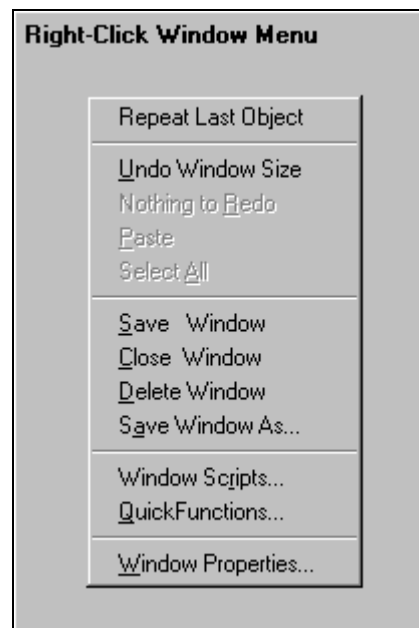
## Working with WindowMaker Windows

Your OpenHMI application will more than likely be comprised of numerous windows that you create to hold your graphics and text objects. When you create a new window in WindowMaker, you will be required to define certain properties for that window such as, its background color, title, screen position and so on. You can also create QuickScripts that execute based upon whether the window is opening, showing or closing.


This section contains the procedures that you will follow to create, open, save, close, delete, and duplicate windows.

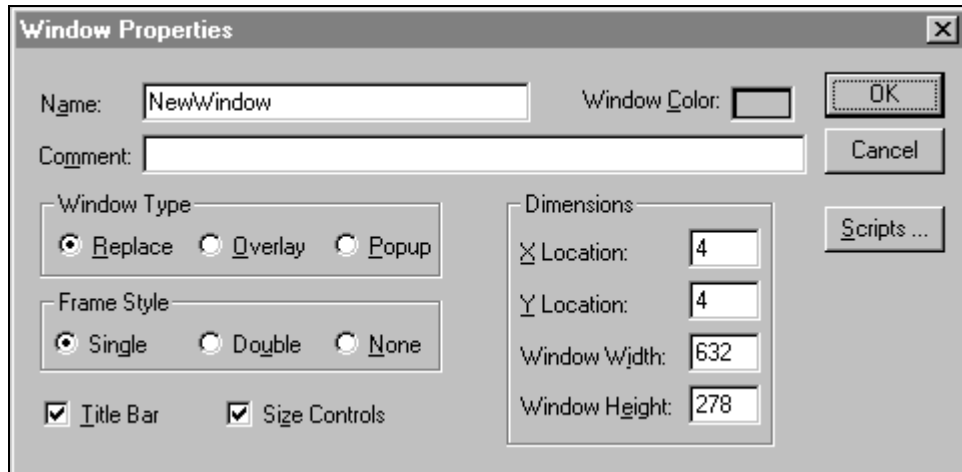
- ☞ The **General Toolbar** contains tools that you can use to quickly apply most of the windows commands on the **File** menu.

To quickly access the various commands that can be applied to a window, right-click a blank area of the open window, and then click the appropriate command on the right-click menu. For example:



## Creating a New Window

-  **To create a new window:**
  1. On the **File** menu, click **New Window** or, click the New Window tool on the **General Toolbar**. The **Window Properties** dialog box appears:
    - ☞ To quickly create a new window, in the Application Explorer, right-click **Windows**, and then click **New**.



**Note** By default, the settings in this dialog box will reflect those of the previously created window or, if you select this command while a window is open in WindowMaker, the settings will reflect those of the active window. If a Window script(s) is attached to the active window, a message box will appear asking you if you want the window script(s) copied to the new window.

- ☞ If you right-click any of the text entry boxes in any dialog box, a menu will appear displaying the commands that you can apply to the selected text.
2. In the **Name** box, type the name that you want to appear in the new window's title bar. The name can be up to 32 characters long. It can include embedded spaces, punctuation marks, and any other character on the keyboard except, quotation marks (").
  3. In the **Comment** box type any miscellaneous comments that you want associated with the window (optional). This information is for documentation purposes only and is not used by the application.
  4. Click the **Window Color** box to select the background color for the window. The color palette will appear:



- ☞ If no change is desired, click on the current color selection or press the ESC key to close the palette.
- ☞ For more information on using the color palette, see Chapter 1 - WindowMaker Program Elements.

5. Click the color that you want to use for the window's background.
6. Select the **Window Type** that you want to use. There are three window types:
  - Replace** Automatically closes any window(s) it intersects when it appears on the screen including popup and other replace type windows.

<b>Overlay</b>	Appears on top of currently displayed window(s) and can be larger than the window(s) it is overlaying. When an overlay window is closed, any window(s) that were hidden behind it will reappear. Clicking on any visible portion of a window behind an overlay window will bring that window to the foreground as the active window.
<b>Popup</b>	Similar to an overlay window except, it always stays on top of all other open windows (even if another window is clicked). Popup windows usually require a response from the user in order to be removed.

---

**Note** You can change a window's type by opening its **Window Properties** dialog box again. There are three ways to do this:

- Open the window, and then on the **Windows** menu, click **Window Properties**. The **Window Properties** dialog box appears.
  - In the Application Explorer under **Windows**, right-click the window name, and then click **Properties**. The **Window Properties** dialog box appears. If the window is not open when you execute the command, it is automatically opened behind the dialog box.
  - Open the window, right-click a blank area of the window, and then click **Window Properties**. The **Window Properties** dialog box appears.
- 

- Select the **Frame Style** for the window. There are three styles:

<b>Single</b>	3-D bordered window that can have a title bar and <b>Size Controls</b> .
<b>Double</b>	3-D bordered window that has no title bar and cannot be sized without <b>Size Controls</b> .
<b>None</b>	A window without a border that cannot be sized without <b>Size Controls</b> . (With <b>Size Controls</b> it becomes a 3-D bordered window that can be sized.)

- Select **Title Bar** if you want the window to have a title bar. The title bar is also used to move the window by clicking the mouse on it, and then dragging the mouse.

☞ If your window has a title bar, you cannot select **Double** or **None** for the **Frame Style**.

- Select **Size Controls** if you want the user to be able to resize the window in WindowViewer.

- In the **Dimensions** group, type the pixel location that you want for each of the window's coordinates:

<b>X Location</b>	The number of pixels between the left edge of the WindowMaker design area and the left edge of the window being defined.
<b>Y Location</b>	The number of pixels between the top edge of the WindowMaker design area and the top edge of the window being defined.

**Window Width** The window's width in pixels.

☞ Windows limits the minimum width of a window according to your display monitor. For example, for the standard VGA, the minimum is 102 pixels.

<b>Window Height</b>	The window's height in pixels.
	☞ Windows limits the minimum height of a window according to your display monitor. For example, for standard VGA, the minimum is 26 pixels.

---

**Note** By default, the values in these boxes will be set to the dimensions of the previously created window. They are also automatically modified if you manually change the windows size in WindowMaker by using the window's border.

---

11. Click **Scripts** to access the **Window Script** editor. There are three types of scripts that you can apply to a window:

<b>On Show</b>	Executes one time when the window is initially shown.
<b>While Showing</b>	Executes continuously at the specified frequency while the window is showing.
<b>On Hide</b>	Executes one time when the window is hidden.

---

**Note** If you attach a Window Script to the active window and then you create a new window, the script(s) from the active window can be copied to the new window. A message dialog box will appear asking if you want to copy the window script(s).

---

- ☞ If you later decide that you need to attach a script to an open window, right-click a blank area of the open window, and then click **Window Scripts**. If the window is not open, in the Application Explorer, double-click **Windows** to show all window names. Right-click the window name, and then click **Window Scripts**.

☞ For more information on creating window scripts, see Chapter 5 - Creating QuickScripts in OpenHMI.

## Creating a Window to Hide the Title and Menu Bars

The WindowMaker design area is the entire area below the title and menu bars and within the window frame. The design area becomes the display area in WindowViewer. The specific location X=0 and Y=0 is always the upper left corner just below the title and menu Bars. The title and menu bars are each 19 pixels high and are above the design area. For example, if WindowMaker is maximized, and you are using a 1024x768 video display, the visible design area would 1024x730 (768 less 19 pixels for the title and 19 pixels for the menu bar equals 730 pixels in the visible design area). If WindowViewer is configured to show its title bar and menu bar, the display area in WindowViewer will fill the screen below the title bar and menu bar exactly as seen in WindowMaker.

To take advantage of the additional space used by the title and menu bars, you can design an application with the title bar and menu bars hidden. When the title bar and menu bar are hidden, the upper left corner of the window references a new position on the screen. This increases the visible display area and provides you with more display area. If you configure WindowViewer in this manner, all of your windows will automatically appear to move up, and a gap will appear at the bottom of the window. To fill this gap, you need to increase the window height by setting the Y location of the window to a negative value. This places the window under the title bar and menu bar in WindowMaker, and on top of the them in WindowViewer.



You can use this technique with a popup window to hide cover the title bar and menu bar in WindowViewer. You can also create a touch link pushbutton or QuickScript to hide this popup window whenever you need to reveal the title bar and menu bar to the user. In addition, by applying security and a password, you can restrict certain users from hiding the window and gaining access to the menus which includes the ability to exit WindowViewer.

---

**Note** A Promotional OpenHMI license does not allow the title bar to be hidden. Therefore, if an application is developed under a Promotional OpenHMI license and WindowViewer is configured with the **Hide Title Bar** option selected when that application is viewed on a standard runtime license, the title bar will be hidden as configured and, all the windows in the application to move up.

---

## Opening Windows

While developing your application, you can open as many windows as your computer memory will support.



### ➤ To open a window(s):

1. On the **File** menu, click **Open Window** or, click the Open Window tool on the **General Toolbar**. The **Windows to Open** dialog box appears listing the names of all windows in your application.
  - ☞ To quickly open a single window, in the Application Explorer, double-click **Windows** to open the list of all the window names in your application, and then double-click the window name. You can also right-click the window name, and then click **Open**.
2. Click the check box next to the name of the window(s) that you want to open.
  - ☞ By default, all currently opened windows will already be checked.
3. Click **OK** to close the dialog box and open the selected window(s).

## Saving Windows

Once you have created a window, you will need to save it before you can close it or exit your application. All graphics, QuickScripts, properties, and so on associated with the window are also saved.



### ➤ To save a window(s):

1. On the **File** menu, click **Save Window** or, click the Save Window tool on the **General Toolbar**. The **Windows to Save** dialog box appears listing the names of all windows that need to be saved.
  - ☞ To quickly save a single window, in the Application Explorer, right-click the window name, and then click **Save**. You can also right-click any blank area of the window, and then click **Save Window**.



To quickly save all currently open windows, click the Save All Windows tool on the **General Toolbar**, or the **Save All** command on the **File** menu.

2. Click the check box next to the name of window(s) that you want to save.

3. Click **OK** to close the dialog box and save the selected window(s).

## Closing Windows

If you attempt to close a window(s) that has been modified since it was last saved, you will be prompted to save your changes before WindowMaker will close the window.



### ➤ To close a window(s):

1. On the **File** menu, click **Close Window** or, click the Close Window tool on the **General Toolbar**. The **Windows to Close** dialog box appears listing the names of all currently open windows.
  - ☞ To quickly close a single window, in the Application Explorer, right-click the window name, and then click **Close**. You can also right-click any blank area of the window, and then click **Close Window**.
2. Click the check box next to the name of the window(s) that you want to close.
3. Click **OK** to close the dialog box and close the selected window(s).

## Deleting Windows

Deleted windows cannot be restored unless you have backed them up. You will be prompted to confirm the deletion of each window name you select.

### ➤ To delete a window(s):

1. On the **File** menu, click **Delete Window**. The **Windows to Delete** dialog box appears listing the names of all currently open windows.
  - ☞ To quickly delete a single window, in the Application Explorer, right-click the window name, and then click **Delete**. You can also right-click any blank area of the window, and then click **Delete Window**.
2. Click the check box next to the name of the window(s) that you want to delete.
3. Click **OK** to close the dialog box and delete the selected window(s).

## Duplicating a Window

When you want to create a duplicate copy of an existing window, the window that you want to duplicate must be open.

### ➤ To duplicate a window:

1. On the **File** menu, click **Save Window As**. The **Window to save under new name** dialog box listing the names of all currently open windows will appear.
  - ☞ To quickly duplicate a window, in the Application Explorer, right-click the window name, and then click **Save As**. You can also right-click in any blank area of the window, and then click **Save Window As**.
2. Click the check box next to the name of window that you want to duplicate. (Only one window name can be selected.) The **Save "*Window Name*" As** dialog box appears:



3. In the **New Name** box, type a valid name for the new window.
4. Click **OK** to close the dialog box and create the duplicate window.

## Exporting Windows

Exporting windows is very useful when you need to create or maintain a library application. To move windows from one OpenHMI application to another, you must use the **Export Window** command on the **File** menu.

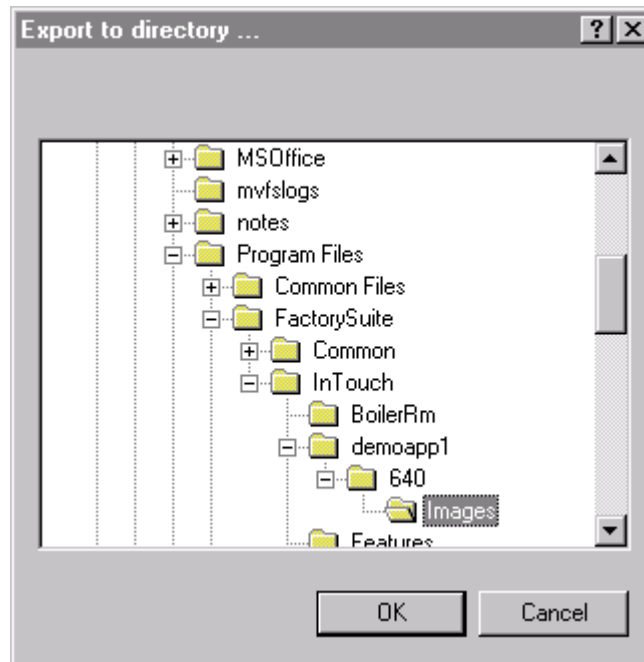
---

**Note** If you attempt to copy OpenHMI window files by using any other copy methods, such as the File Manager or Windows Explorer copy commands, you may corrupt your application's Tagname Dictionary!

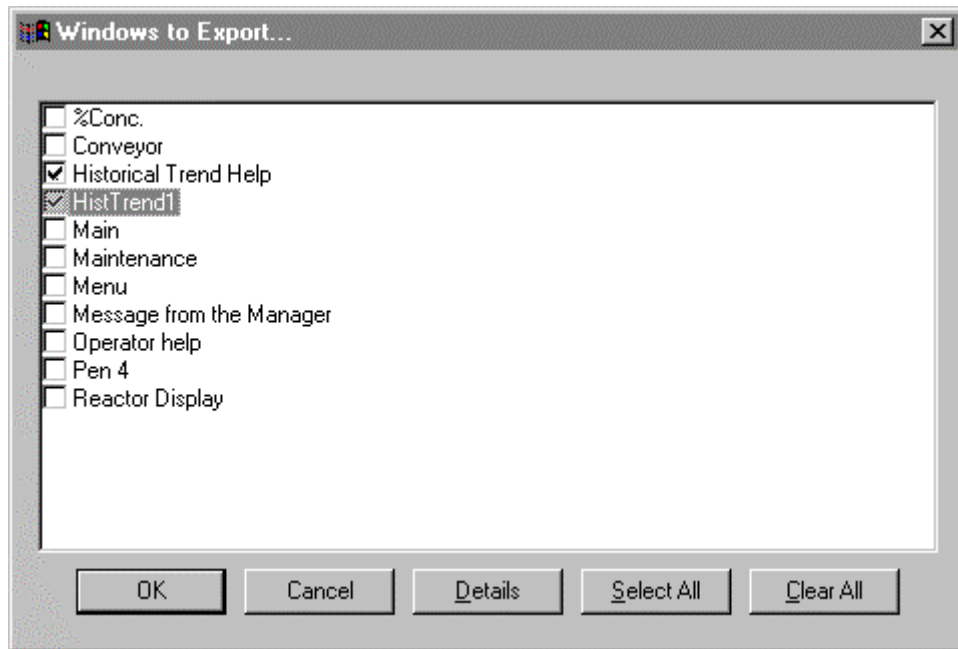
---

➤ **To export a window:**

1. Close all windows in your current application.
2. On the **File** menu, click **Export Window**. The **Export to directory** dialog box appears:



3. Locate and select the application directory (folder) that you want to export the window(s).
  - Click **Yes** if you want to export the window.
  - Click **No** if you want to export the window(s) to another OpenHMI application.
4. The **Windows to Export** dialog box appears:



5. Select the window(s) that you want to export.
6. Click **OK**. The export operation will take place.

---

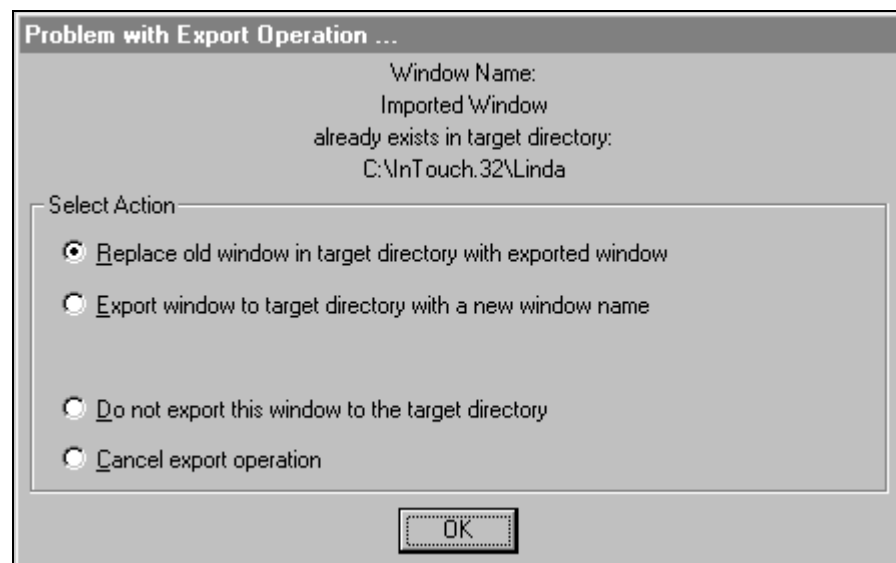
**Note** When you export a window, all of the objects and animation links associated with that window are exported with the window. However, the tagnames associated with the objects in the window are converted into "placeholder" tagnames. Placeholder tagnames are used to avoid any problems that might occur when the destination application's Tagname Dictionary does not contain the same tagnames.

---

↪ For more information on converting placeholder tagnames, see Chapter 3 - Tagname Dictionary.

## Problem with Export Operation

If a problem is encountered by the system when exporting the window, the **Problem with Export Operation** dialog box appears:



In the **Select Action** group, select the action that you want to take, and then click **OK**.

## Importing Windows

Importing windows from one OpenHMI application to your current application, can save you a considerable amount of development time. It allows you to reuse your previously created windows, objects and window scripts. When you move windows from one OpenHMI application to another, you must use the **Import** command on the **File** menu.

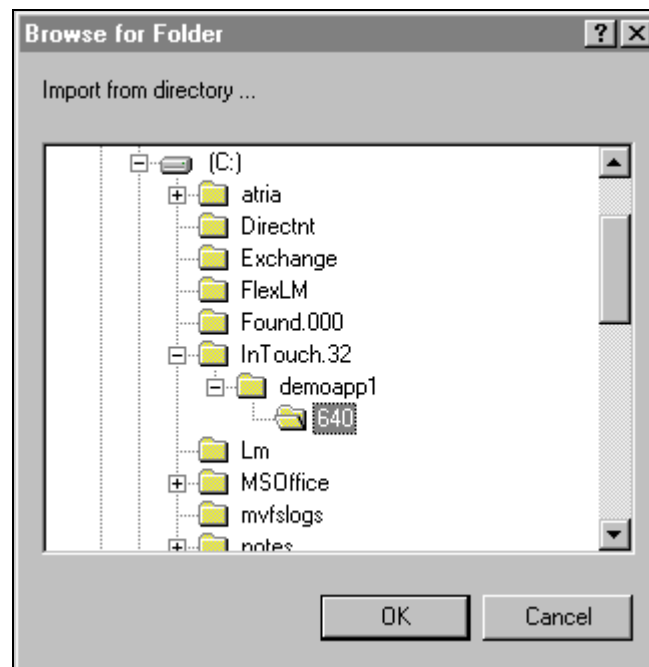
---

**Note** If you attempt to move OpenHMI window files by using any other move methods, such as the File Manager or Windows Explorer move commands, you may corrupt your application's Tagname Dictionary!

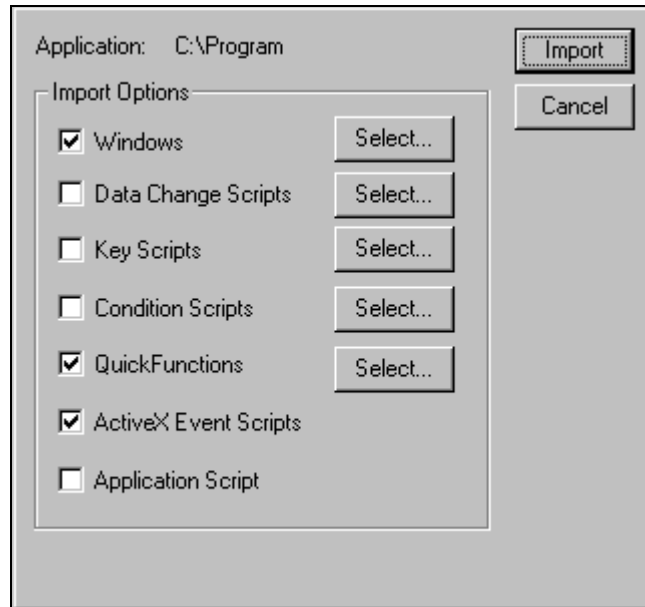
---

➤ **To import a window or QuickScript:**

1. Close all windows in your current application
2. On the **File** menu, click **Import**. The **Browse for Folder** dialog box appears:



3. Locate and select the application directory (folder) containing the windows that you want to import.
4. Click **OK**. The following dialog box appears:



5. Select the item(s) that you want to import, and then click **Select**. A dialog box will appear for you to select the window or QuickScript(s) that you want to import.
6. Once you have selected the window(s) or QuickScript(s) that you want to import, click **Import**. The system will automatically begin to import the selected items into your current application.

---

**Note** To import a window script, you must import the entire window. When you import a window, all of the objects and links associated with that window are imported with the window. However, the tagnames associated with the objects in the window (and the tagnames used in an imported script) are converted into "placeholder" tagnames.

When the tagnames in an imported script or window are converted to placeholder tagnames, three index characters are added to the beginning of each tagname. For example, when a discrete tagname is imported, the tagname is prefixed with the three characters **?d:**. When a tagname of 30, 31 or 32 characters in length is imported, the three indexing characters will still be added to the beginning of each tagname. However, the addition of these three characters will not truncate the length of your existing tagname. For example, for placeholder tagnames only, a 32 character tagname is increased to 35 characters. These three additional spaces are allotted only for placeholder tagnames. This increase in tagname length is not supported for standard tagnames.

---

For more information on converting placeholder tagnames, see Chapter 3 - Tagname Dictionary.

## Working with Graphic Objects

Once you have created a new window in your application, you are ready to populate it with graphic objects. WindowMaker provides you with numerous tools for editing and arranging the various graphic objects that you will draw and paste into your windows. This section describes the procedures you will use to perform various editing functions on your graphic objects.

- ☞ The **Draw Object Toolbar** contains the graphic drawing tools that you will use to create graphics in your windows.

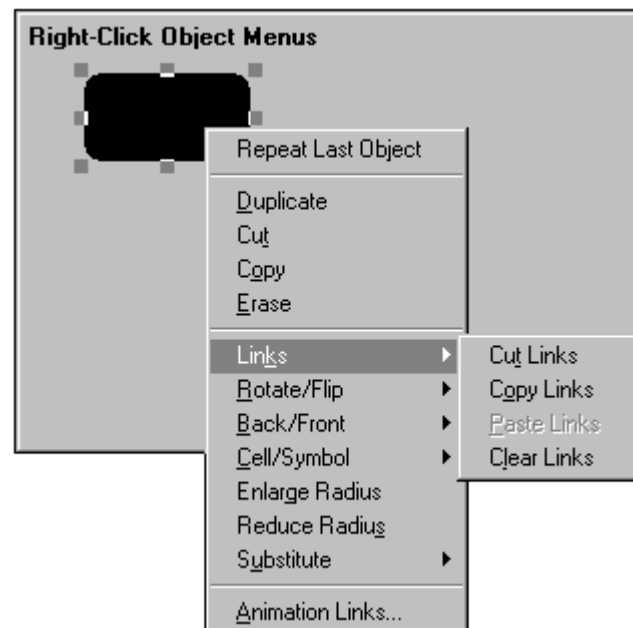
The **View Toolbar** contains the Ruler tool that you can use to display a ruler for assisting you in alignment of your graphic objects in your windows. This toolbar also contains the tools that you will use to hide or show the Application Explorer, the toolbars, or a visible grid in your windows. It also contains the tool for switching to and from full screen mode.

The **General Toolbar** and the **Format Toolbar** contain the tools that you can click to quickly apply many of the commands on the **Edit** menu and the **Text** menu to your graphic objects.

The **Arrange Toolbar** contains the tools that you can click to quickly apply the alignment commands on the **Arrange** menu.

You can also customize the **Wizards/ActiveX Toolbar** by adding any wizards or ActiveX controls that you repeatedly use.

- ☞ For more information on the WindowMaker toolbars, see Chapter 1 - WindowMaker Program Elements.
- ☞ If you right-click a object, a menu will appear displaying the valid commands or actions that you can apply to the selected object. For example:




---

**Note** The commands on the right-click menus will vary. They are based upon the type of object that is selected.

---

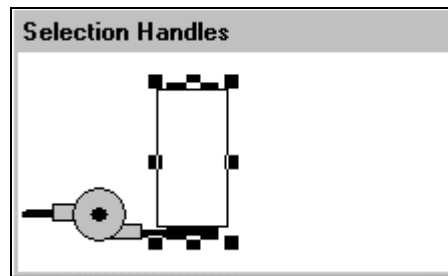
## Selecting and Sizing Objects

After you draw an object, and you click it, several little boxes will surround it. These boxes are called "handles." You use these handles to resize and/or reshape the object.

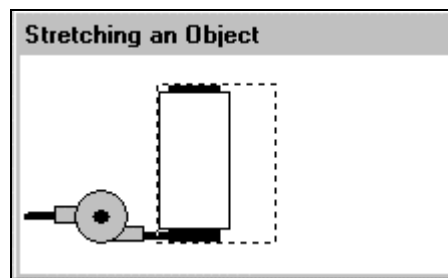
The notion of "selected" is a key concept of WindowMaker graphics editing. The presence of "handles" around an object indicates that it is "selected." Clicking directly on an object selects it. Clicking on a blank area of the window deselects any currently selected object(s) in that window. In general, any command that you execute is applied to all selected objects, if the command is valid for the object.

### ➤ To change the size of an object:

1. Select the object, and then position the point of the arrow cursor in the center of a "handle."



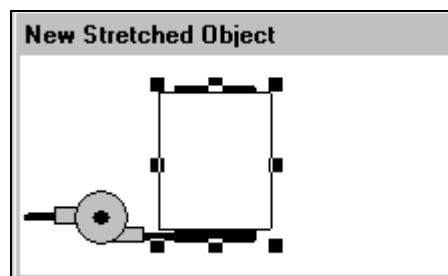
2. Press and hold down the left mouse button while you drag the handle to either stretch or shrink the object:




☞ You can stretch/shrink the object in either direction, depending upon which handle is selected. If you use one of the four corner handles to size the object, it will be sized vertically and horizontally simultaneously.

3. Release the

mouse button. The object will be redrawn at its new size:



- ☞  If the size is not correct, on the **Edit** menu, click **Undo** or, click the Undo tool on the **General Toolbar** and try again.

☞ For more information on the WindowMaker toolbars, see Chapter 1 - WindowMaker Program Elements.

☞ For more information on undoing and redoing edits, see "[Undoing Object Edits.](#)"



## Selecting all Objects in a Window

To select all objects in the active window, on the **Edit** menu, click **Select All** or, press F2 or, right-click a blank area of the open window, and then click **Select All**.

## Selecting Multiple Objects

To select multiple objects, select your first object, then hold down the SHIFT key, while you click the other objects that you want to select. To deselect a specific object from a group of selected objects, while all objects are selected, hold down the SHIFT key and click the object that you want to deselect.

## Selecting a Group of Objects

Move the cursor to a blank area of your window. Depress the mouse button and drag the mouse. A dotted selection rectangle with a small hand cursor will appear. Drag the mouse until you have completely surrounded all of the objects that you want to select. Release the mouse button. All the objects that were completely within the rectangle will be selected.

## Deselecting a Group of Selected Objects

If you hold the SHIFT key down while you draw a selection rectangle, all enclosed selected objects will become deselected. This technique may also be used to start a selection rectangle on top of another object.

If you hold the SHIFT key down while you click the left mouse button, the object under the cursor will not be dragged when the cursor is moved. Instead, a selection rectangle will be drawn.


## Undoing Object Edits

WindowMaker keeps track of the editing and formatting changes that you make. You can configure WindowMaker to support up to 25 undo/redo levels. You can also disable the undo/redo functionality by setting the level to zero (0). By default, WindowMaker is set to support 10 levels where, one level represents one action.

---

**Note** The undo and redo stacks are empty when you create a new window or open an existing window. Both stacks are also cleared when you save the window.

---

 For more information on configuring the undo/redo levels, see "[Customizing Your Development Environment](#)."

The **Undo** and **Redo** commands are located on the **Edit** menu. These commands can also be accessed by right-clicking the object. The commands are dynamic and will change to reflect the last action to which they can be applied. For example, if you move an object then decide that you want to return it to its original location in the window, right-click a blank area of the window and click **Undo Move** or, on the **Edit** menu click **Undo Move**.



The Undo and Redo tools are located on the **General Toolbar**.


You will use undo and redo to reverse actions or commands that you have applied to an object. You can undo or redo the following actions or commands:

Action/Command	Supported
Basic	Create, Select, De-select, Move and Re-size Object Line, Fill, Text and Window Color

	Window Move and Window Size
<b>Editing</b>	Duplicate, Cut, Copy, Paste, Delete Paste Bitmap and Adjust Bitmap - Original size Select All, Link Cut, Link Copy, Link Paste, Link Clear Enlarge Radius, Reduce Radius Reshape Object, Add Point, Delete Point
<b>Arranging</b>	Send to Back, Bring to Front, Align (all commands) Space Horizontal, Space Vertical, Rotate CW, Rotate CCW Flip Horizontal, Flip Vertical Make Symbol, Break Symbol Make Cell, Break Cell
<b>Text</b>	All operations (size, style, pitch, justification)
<b>Line</b>	All operations (width and styles)
<b>Special</b>	Animation Links (double-click on object), Substitute Strings

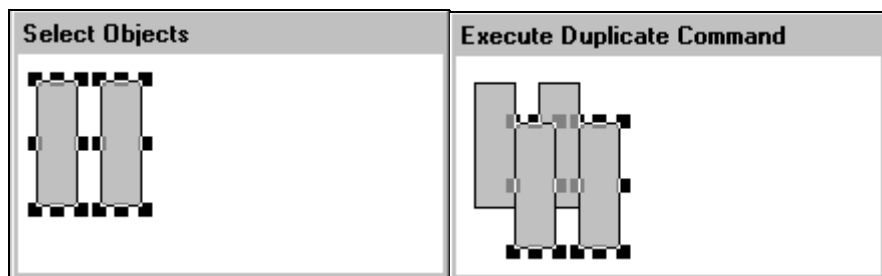
## Duplicating Objects

- **To redraw the previously drawn object:**
  1. Draw the object.
  2. Right-click the object, and then click **Repeat Last Object**.
  3. Click the left mouse button, and then draw the same object again.

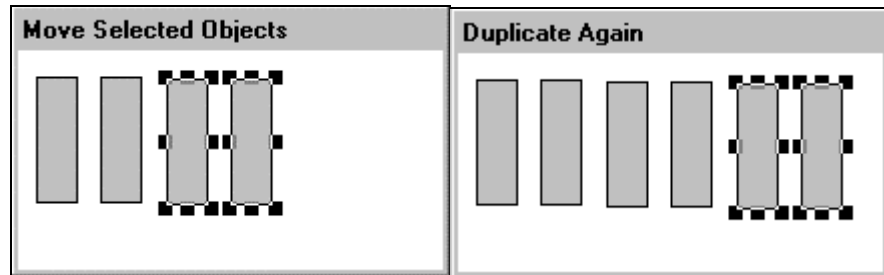
-  **To duplicate an object:**
  1. Select the object(s) you want to duplicate.
  2. On the **Edit** menu, click **Duplicate** or, click the Duplicate Selection(s) tool on the **General Toolbar**.

*ℹ* For more information on the WindowMaker toolbars, see Chapter 1 - WindowMaker Program Elements.

- ☞ To quickly duplicate an object(s), right-click the object, and then click **Duplicate**.

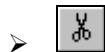


- ☞ If you move the duplicated object(s) without deselecting it, and you duplicate it again, the second (and all subsequent duplications) will automatically be offset the same distance that the first duplicate was moved. For example:



☞ You can repeat this procedure as many times as necessary.

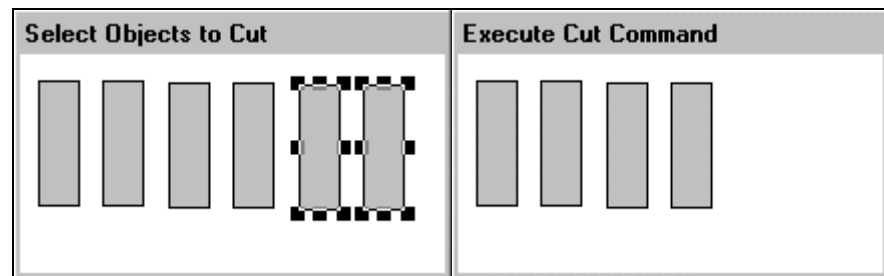
## Cutting Objects to the Windows Clipboard



➤ **To cut an object:**

1. Select the object(s) you want to cut.
2. On the **Edit** menu, click **Cut** or, click the Cut to Clipboard tool on the **General Toolbar**.

☞ To quickly cut an object(s), right-click the object, and then click **Cut**.



**Note** When you cut an object, it is erased from your window and copied to the Windows Clipboard. The object's attributes and animation links are also copied with it.

## Copying Objects to the Windows Clipboard

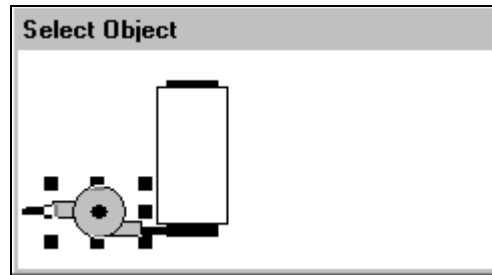


➤ **To copy an object:**

1. Select the object(s) you want to copy.
2. On the **Edit** menu click **Copy** or, click the Copy to Clipboard tool on the **General Toolbar**.


☞ For more information on the WindowMaker toolbars, see Chapter 1 - WindowMaker Program Elements.

☞ To quickly copy an object(s), right-click the object, and then click **Copy**.

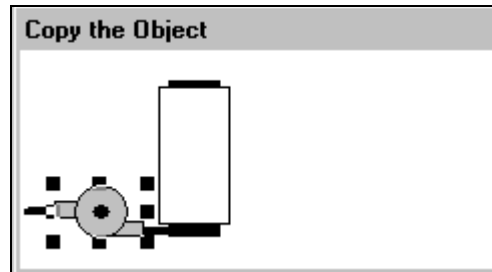


**Note** When you copy an object, it is not erased from your window. It is copied to the Windows Clipboard. The object's attributes and animation links are also copied with it.

## Pasting Objects from the Windows Clipboard

-  To paste an object from the Windows Clipboard:

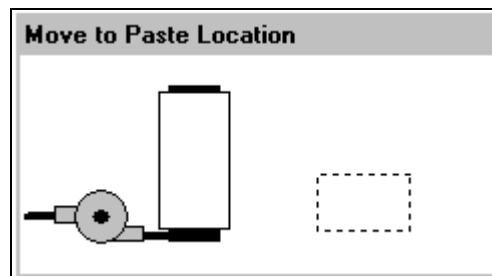
1. Copy or cut the object:



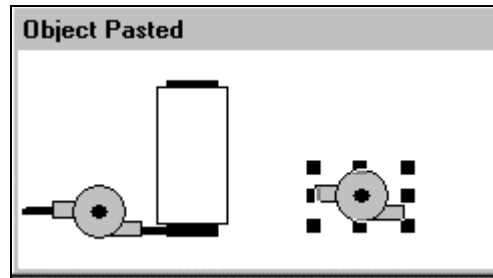
2. On the **Edit** menu, click **Paste** or, click the Paste from Clipboard tool on the **General Toolbar**.

☞ To quickly paste the object, right-click the object, and then click **Paste**.

3. The cursor will change to a corner symbol.
4. Hold down the left mouse button, a dotted line rectangle the size of the copied object will appear. Drag the rectangle to the location in the window that you want to paste the object:



5. Release the mouse button to paste the object:



- ☞ All pasted objects remain selected after being pasted and you can move them to adjust their location.

---

**Note** When you copy or cut an object to the Windows Clipboard, all the object's attributes and animation links are copied with it. If you subsequently paste that object into a window, all of its attributes will still be intact.

---

## Cutting and Pasting Object Links

WindowMaker's link paste buffer is a temporary storage area that stores links that you cut or copy from an object. (The buffer only stores the links for your most recent cut or copy action.) You can paste the links stored in the link paste buffer to any object or symbol. If you select multiple objects, the links are pasted to each individual object.

If a pasted link has no apparent value to the object, for example, a line color link on a text object, the link is not pasted.

☞ For more information on animation links, see Chapter 4 - Creating Animation Links.

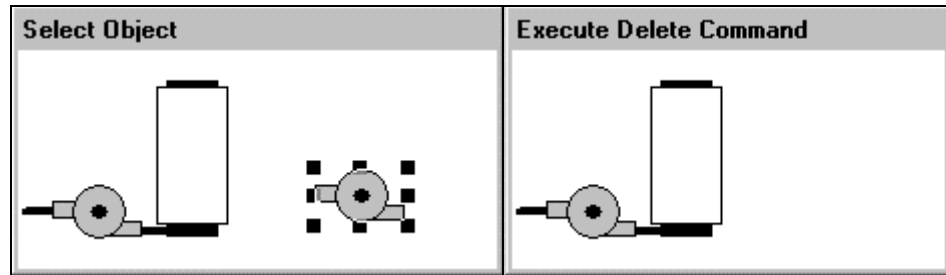
➤ **To cut, copy, paste and clear links:**

1. Select the object that you want to apply the links command to.
2. On the **Edit** menu, point to **Links**, and then click the appropriate command.
  - ☞ To quickly access the link commands, right-click the object, then point to **Links**, and then click the appropriate links command.

## Deleting Objects

➤ **To delete an object:**

1. Select the object(s) you want to delete.
2. On the **Edit** menu, click **Erase**.
  - ☞ To quickly delete an object(s), right-click the object, and then click **Erase**, or select the object and press the DEL key.

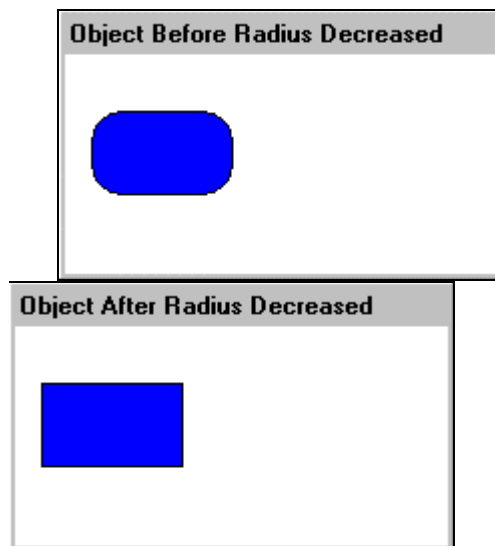


**Note** Deleted objects are not copied to the Windows Clipboard.

## Increasing or Decreasing a Rounded Object's Radius

You can increase and/or decrease the corner radius of any object that you have drawn with the Rounded Rectangle tool.

- **To increase (or decrease) a rounded object's radius:**
  1. Select the object.
  2. On the **Edit** menu, click **Enlarge Radius** (or **Reduce Radius**).
    - ☞ To quickly increase or decrease the radius, right-click the rounded object, and then click the appropriate command.
  3. Repeat the command until the radius has increased to the desired degree. For example:

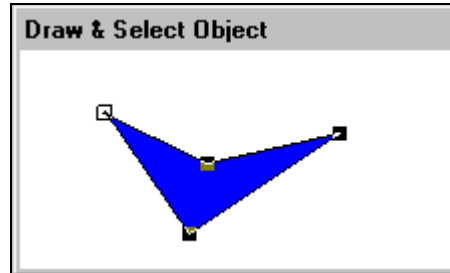


- ☞ You can also use the keyboard short cut keys Shift+Plus, ( + symbol on the numeric keypad), to increase the radius or, Shift+Minus, ( - symbol on the numeric keypad) to decrease the radius. Each time you press these key combinations, the command will be performed. If you hold the keys down, the command will execute continuously until the maximum increased or decreased radius for the object is reached.

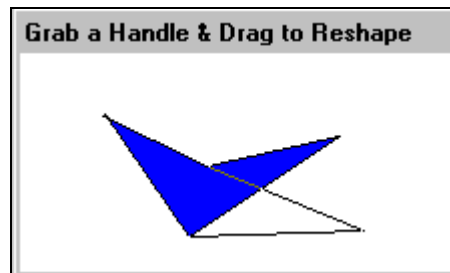
## Reshaping a Polyline or Polygon Object

-  **To adjust the shape of polylines or polygons:**

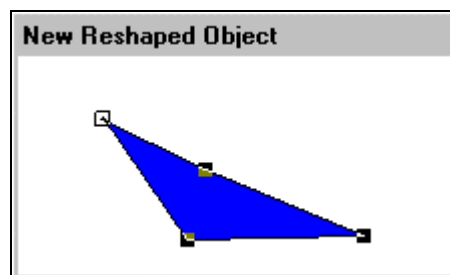
1. Select the polyline or polygon object.



- ☞ Each "point" where you clicked the mouse as you were drawing the object will reappear as a "handle."
2. On the **Edit** menu, click **Reshape Object** or, click the Reshape Object tool on the **Arrange Toolbar**.
    - ☞ To quickly reshape a polyline or polygon, right-click the object, and then click the appropriate command.
  3. To reshape the object, grab a handle and drag it to the desired location:

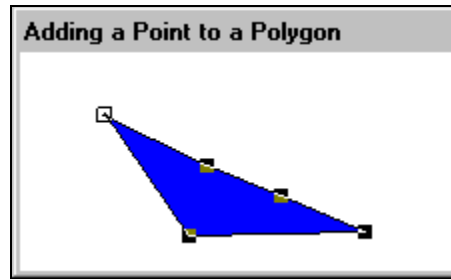


4. When you release the mouse, the object will be redrawn to its new shape:



- **To add or delete "points" on a polygon or polyline:**

1. Select the polyline or polygon object.
2. On the **Edit** menu, click **Add Point** or **Del Point** then click the spot of the object where you want the new point added or, click the point that you want to delete.
  - ☞ To quickly add (or delete) points on a polyline or polygon, right-click the object, and then click the appropriate command.







## Arranging Objects in your Window

WindowMaker provides you with numerous tools that you will use to arrange your objects in your windows. This section describes the various arranging tools that are available in WindowMaker.

- ☞ The **Arrange Toolbar** contains tools that you can use to quickly apply most of the commands found on the **Arrange** menu to selected objects. For example:



- ☞ For more information on the **Arrange Toolbar**, see Chapter 1 - WindowMaker Program Elements.

## Aligning Objects

You can align objects by their left or right edges, centers, centerpoints, tops, middles or bottoms.

### ➤ To align all selected objects:

1. Select the object(s).
2. On the **Arrange** menu, point to **Align**, and then click the appropriate align command. The selected object(s) will be aligned according to your selection.

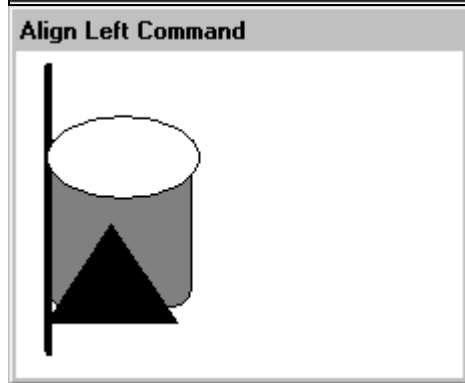
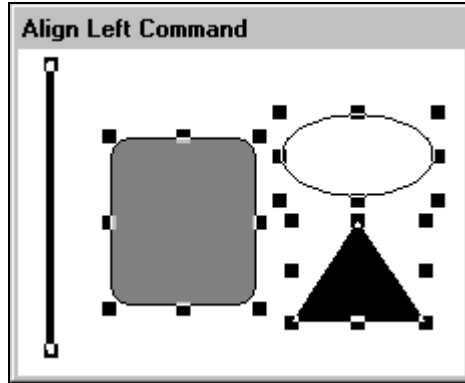
- ☞ To quickly align objects, select the objects, and then click the appropriate tool on the **Arrange Toolbar**.


☞ For more information on the toolbars, see Chapter 1 - WindowMaker Program Elements.

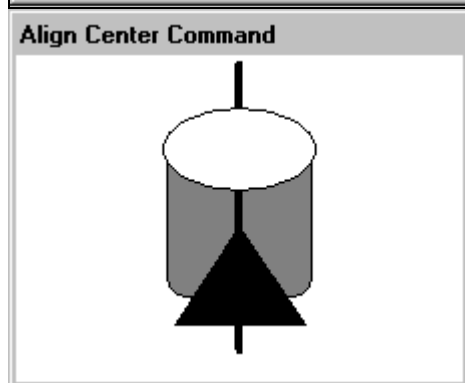
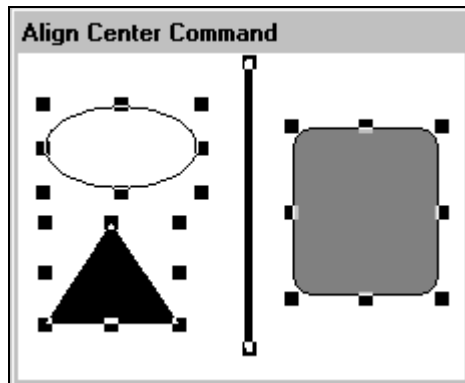
The following examples illustrate the behavior for each alignment command:



**Align Left** aligns the left edge of all selected objects with the left edge of the object that is farthest left in the group:

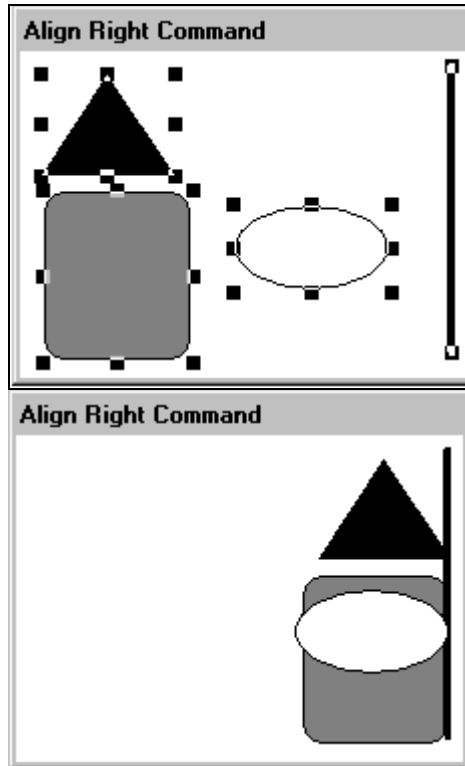


 **Align Center** aligns all the selected objects with the vertical centerline of the group:

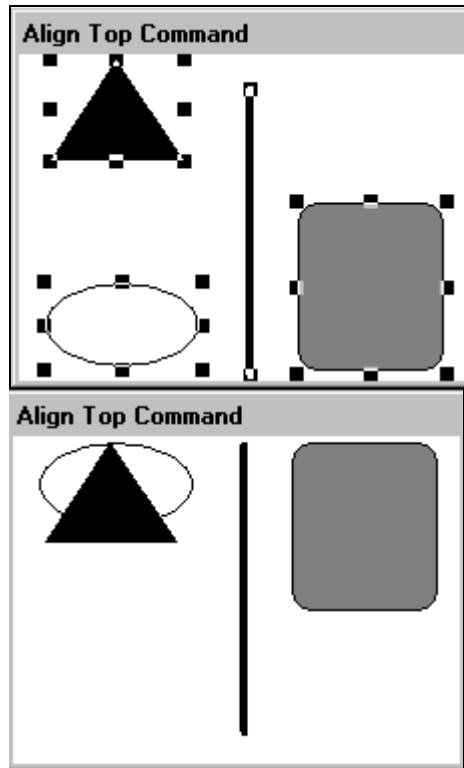




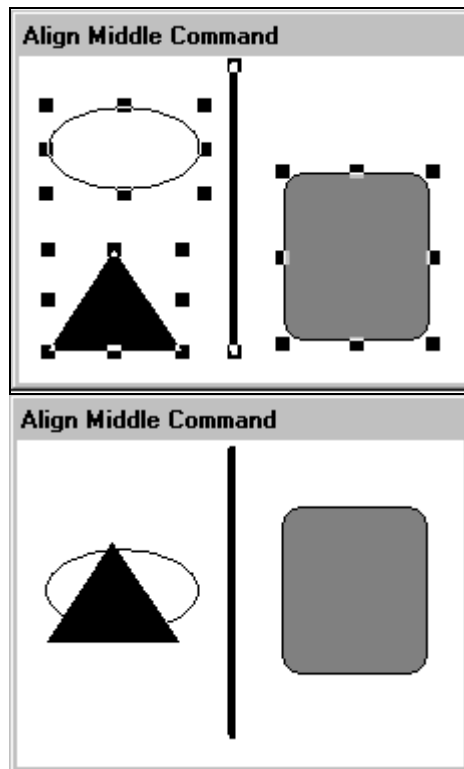
**Align Right** aligns the right edge of all selected objects with the right edge of the object that is farthest right in the group:



**Align Top** the top edge of all selected objects with the top edge of the object that is highest in the group:

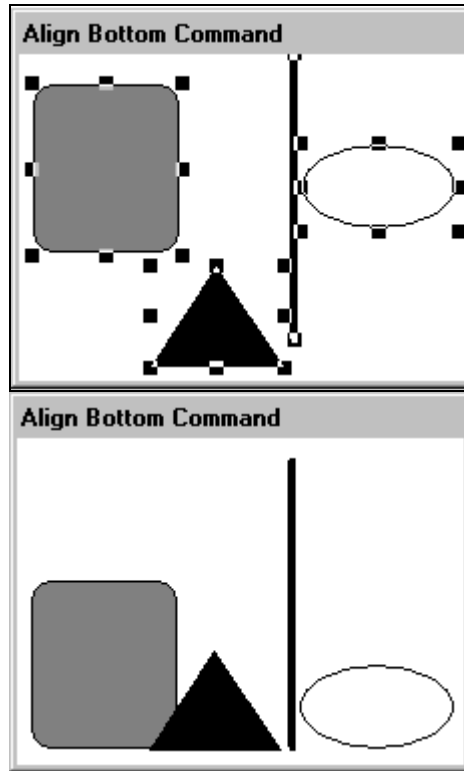


**Align Middle** aligns the middle of all the selected objects with the middle of the group:

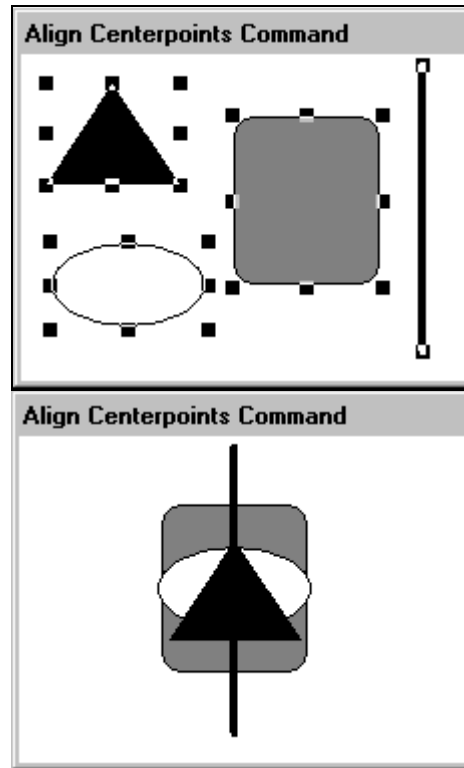




**Align Bottom** the bottom edge of all selected objects with the bottom edge of the object that is lowest in the group:




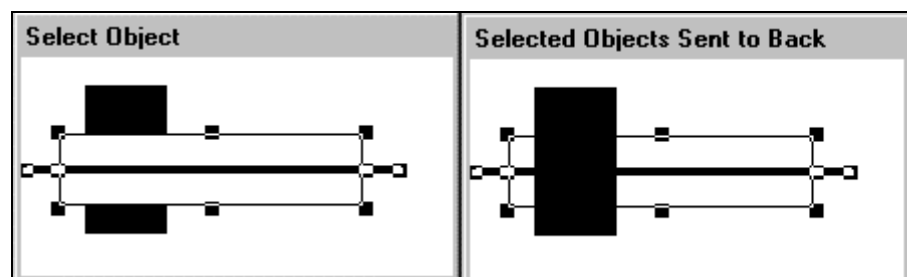
**Align Centerpoints** aligns the centerpoints of all the selected objects with the centerpoint of the group:




## Layering Objects

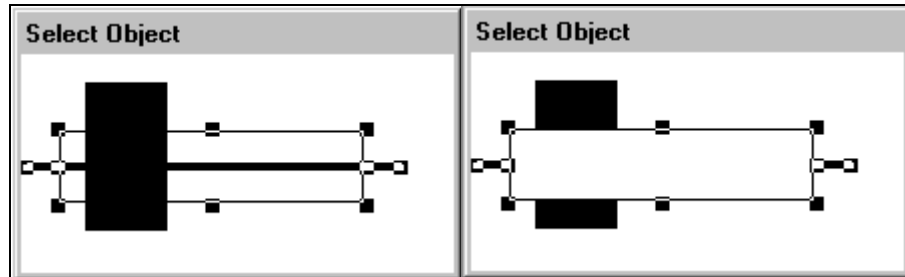
You can layer the objects in your window by positioning objects in front or behind each other.

-  **To position an object behind another object:**
  1. Select the object(s).
  2. On the **Arrange** menu, click **Send to Back** or, click the Send to Back tool on the **Arrange Toolbar**. The selected object(s) will be redrawn behind the object(s) not selected in your window:
    - ☞ To quickly send an object(s) to the back, right-click the object, then point to **Front/Back**, and then click **Send to Back**.



-  **To position an object in front of another object:**
  1. Select the object(s).



- On the **Arrange** menu, click **Bring to Front**, or click the Bring to Front tool on the **Arrange Toolbar**. The selected object(s) will be redrawn in front of the object(s) not selected in your window:
  - To quickly bring an object(s) to the front, right-click the object, then point to **Front/Back**, and then click **Bring to Front**.



Notice that the farthest back object came to the very front.

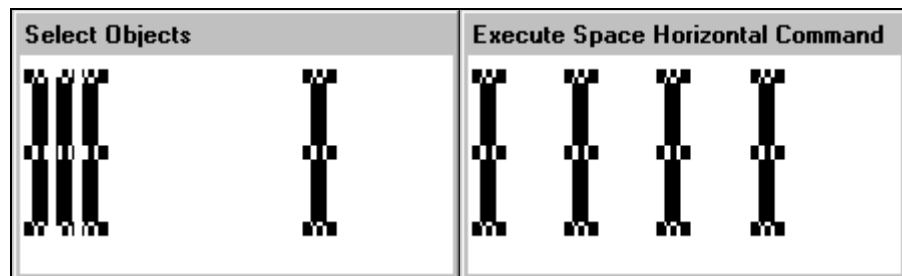
## Controlling Horizontal and Vertical Spacing

You can evenly space objects horizontally between the left most selected object and the right most selected object. You can also control the vertical spacing between the top most selected object and the bottom most selected object.

➤   **To evenly space objects horizontally or vertically:**

- Select the objects.
- On the **Arrange** menu, click **Space Horizontally** (or **Space Vertically**) or, click the appropriate tool on the **Arrange Toolbar**. The selected object(s) will be redrawn spaced evenly between the two farthest placed objects. For example:

For more information on the WindowMaker toolbars, see Chapter 1 - WindowMaker Program Elements.




## Rotating Objects

In WindowMaker, you can rotate most objects including bitmaps, JPEG, PCX, and TGA images and text objects. Objects can be rotated clockwise or counter clockwise 360 degrees in 90 degree increments (90 degrees, 180 degrees, 270 degrees and 360 degrees). Any links attached to the object are rotated with the object. You cannot rotate cells. However, you can rotate symbols.

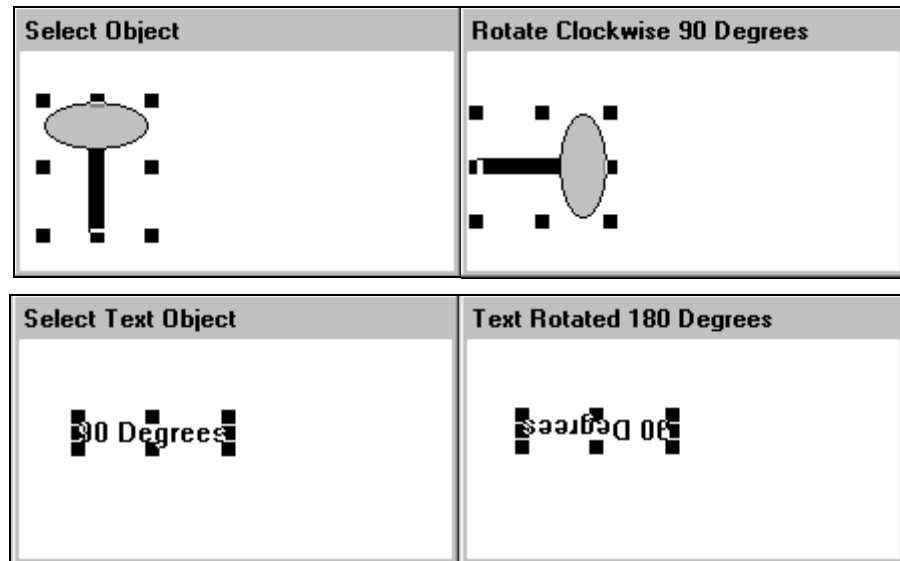
**Note** Rotating objects in WindowMaker has nothing to do with dynamically rotating objects in runtime. Objects are rotated in WindowViewer by linking them to an **Orientation** animation link. Text objects cannot be rotated in WindowViewer.



However, bitmaps or images can be rotated by assigning an **Orientation** animation link to them.

➤  **To rotate a selected object(s) 90 degrees:**

1. Select the object(s).
2. On the **Arrange** menu, click **Rotate Clockwise** (or **Rotate CounterClockwise**). The selected object(s) will be rotated 90 degrees in the direction you chose:
  - ☞ To quickly rotate an object, right-click the object, then point to **Rotate/Flip**, and then click the appropriate command.



- ☞ To rotate an object 180 degrees, repeat this procedure. To rotate an object 270 degrees, perform this procedure twice, and so on.

## Mirroring Objects

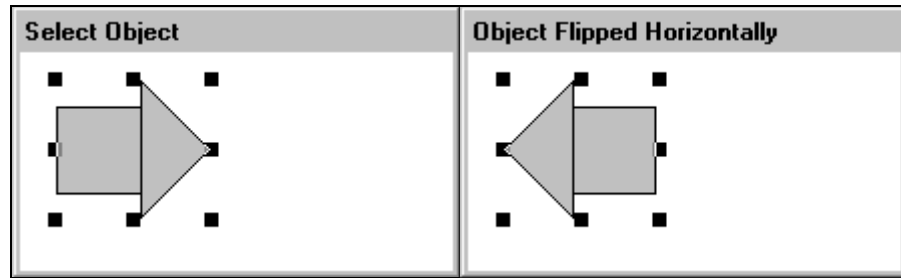
You can flip most WindowMaker objects horizontally or vertically including, bitmaps, JPEG, PCX, and TGA images (text cannot be flipped, it can only be rotated). When you flip an object, you transform it into its horizontal or vertical mirror image. Any links attached to the object are flipped with the object.

➤  **To flip a selected object:**

1. Select the object(s).
2. On the **Arrange** menu, click **Flip Horizontal** (or **Flip Vertical**) or, select the appropriate tool on the **Arrange Toolbar**. The selected object(s) will flip. For example:

☞ For more information on toolbars, see Chapter 1 - WindowMaker Program Elements.

- ☞ To quickly flip an object(s), right-click the object, then point to **Rotate/Flip**, and then click the appropriate command.



## Creating Cells and Symbols

You can combine multiple objects into two different types of single units: cells and symbols. Multiple cells can be combined into a single cell. Cells are objects that maintain a fixed spatial relationship between individual graphic elements. The individual components within a cell, except another cell, can be animated. Cells cannot be resized, nor can animation links be attached to cells. However, animation links can be attached to symbols, and symbols can be included in a cell. All animation links associated with a symbol or an object(s) within a cell are unchanged. The attributes of objects such as text, font, line width, radius and relative positions within a cell cannot be sized until the cell is broken into its individual components.

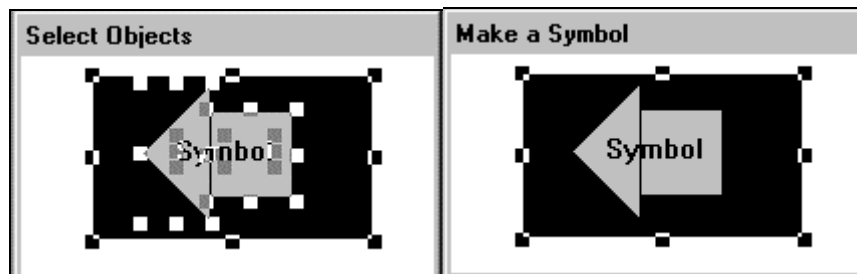
- ☞ Double-clicking a cell will cause the **Substitute Tagnames** dialog box to appear (not the animation links selection dialog box as with objects and symbols).
- ☞ For more information on substituting tagnames, see Chapter 3 - Tagname Dictionary.

---

**Note** When combining cells each cell will be retained, so when the combined cell is broken the original cells are restored.

---

A symbol can be made up of multiple symbols and/or multiple simple objects as illustrated below:



If one of the objects selected has animation links attached to it, the links will be attached to the new symbol. (If the link paste buffer has links in it, you will be asked if you want to paste the links on the new symbol.)

---

**Note** You cannot make a symbol if more than one of the selected objects has links. If you combine two symbols into a new symbol, the original symbol structure is lost. Therefore, if you break the new symbol, it will be broken into the individual components of each original symbol. The two original symbols are lost.

---

- ☞ For more information on animation links, see Chapter 4 - Creating Animation Links.



### To create a symbol or cell:

1. Select the objects that you want to include in the cell or symbol...
2. On the **Arrange** menu, click **Make Cell** (or **Make Symbol**), or click the appropriate tool on the **Arrange Toolbar**.

For more information on the WindowMaker toolbars, see Chapter 1 - WindowMaker Program Elements.

- To quickly create a cell or symbol, select all the objects. Right-click one of the selected objects, point to **Cell/Symbol**, and then click the appropriate command.



### To break a symbol or cell:

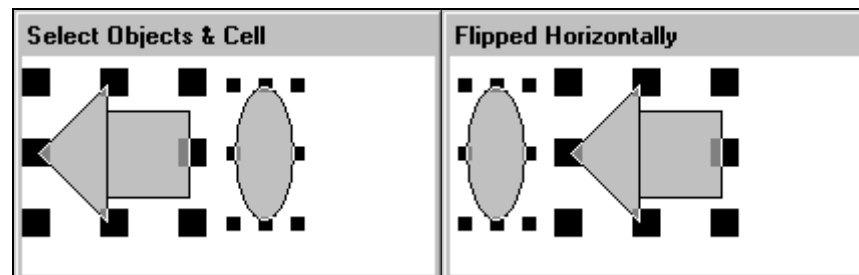
1. Select the symbol or cell...
2. On the **Arrange** menu, click **Break Cell** (or **Break Symbol**), or click the appropriate tool on the **Arrange** toolbar.

For more information on the WindowMaker toolbars, see Chapter 1 - WindowMaker Program Elements.

- If the symbol has links defined for it, the links are automatically saved to the link paste buffer.
- To quickly break a cell or symbol, right-click the cell or symbol, then point to **Cell/Symbol**, and then click the appropriate command.

## Flipping Cells

When you flip cells, they are not mirrored. Only the position of the cell in the group of objects is mirrored. For example:



Compare the location of the cell (on the left) and direction it is facing before it is flipped to the direction it is facing after it is flipped. Its position was flipped, but not its contents. The ellipse object was mirrored. The same applies to cells flipped vertically.

## Snapping Objects to the Grid

When you are arranging objects in your windows, turning on the grid will cause your graphic to snap at the upper left pixel interval on the grid. If you select multiple objects, the snapping will be applied to the upper left corner of the first object selected in the group.

- By default, the grid is set to 10 pixels and visible when you initially start WindowMaker. You can configure the pixel interval for the grid through the **WindowMaker Properties** dialog box.



Click the Snap to Grid tool on the **View** toolbar to turn snap to grid on and off, or on the **Arrange** menu, click **Snap to Grid**.

For more information on the WindowMaker toolbars, see Chapter 1 - WindowMaker Program Elements.

➤ **To configure the grid:**

1. On the **Special** menu, point to **Configure** then click **WindowMaker** or, in the Application Explorer under **Configure**, double-click **WindowMaker**. The **WindowMaker Properties** dialog box will appear.
2. In the **Spacing** box, type the number of pixels that you want spaced between the snap to grid's coordinates.
3. Select **Show Grid** if you want a visible grid in your windows when you turn on WindowMaker's "snap to grid" functionality.

If you do not select **Show Grid**, no grid will be visible in your windows when you turn snap to grid on.

For more information on configuring the grid, see "[Customizing Your Development Environment](#)."

## Working with Images and Bitmaps

All graphic objects such as pictures, screen captures, AutoCad drawings, JPEG, PCX and TGA file types and so on, that are created in other Windows programs must be pasted into a bitmap container in WindowMaker.

WindowMaker sees a bitmap as a single object. Therefore, you cannot animate the individual elements of a bitmap, nor can you include bitmaps in symbols. However, you can include them in a cell.

In WindowMaker, you can rotate bitmaps, JPEG, PCX, and TGA images. They can be rotated clockwise or counter clockwise 360 degrees in 90 degree increments (90 degrees, 180 degrees, 270 degrees and 360 degrees). Any links attached to the bitmap are rotated with it.


---

**Note** Rotating bitmaps in WindowMaker has nothing to do with dynamically rotating them in runtime. Bitmaps or images are rotated in WindowViewer by linking them to an **Orientation** animation link.

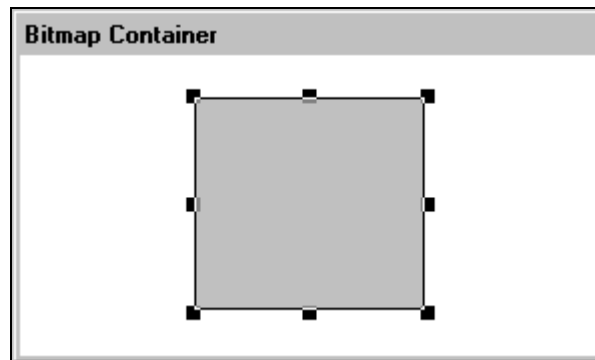
---

You can also define a bitmap with a transparent color, so that you can float it over other objects. When you define a bitmap with a transparent color, the window background color or any objects behind the bitmap will show through it everywhere the transparent color is used. (Only one transparent color may be used per bitmap.)

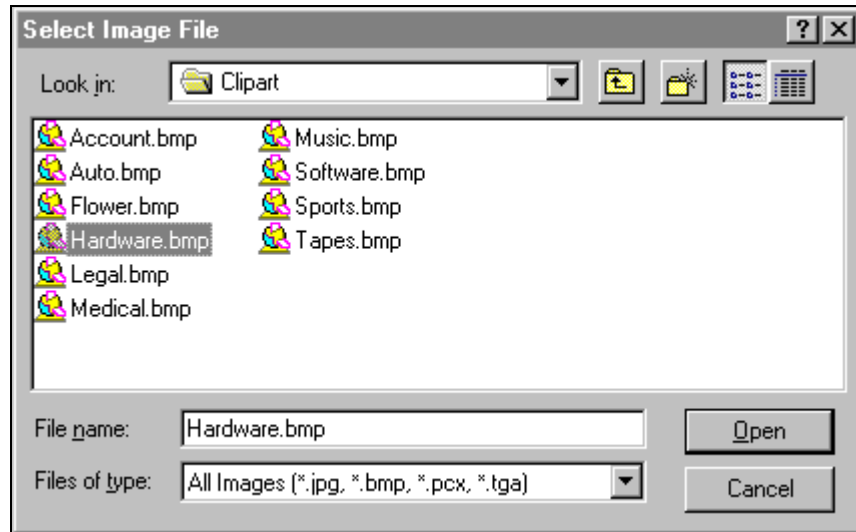
For more information on transparent bitmaps and images, see "[Creating a Transparent Bitmap.](#)"

➤  **To import a bitmap or JPEG, PCX or TGA file type:**

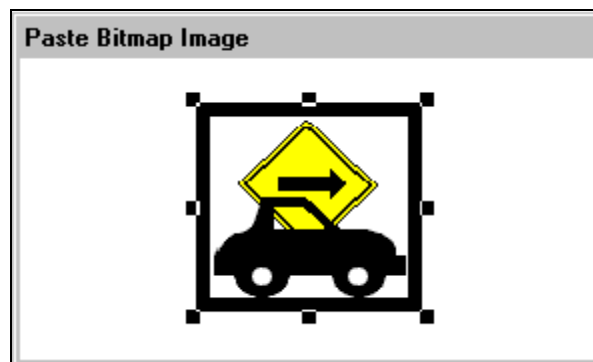
1. Click the Bitmap tool (your cursor turns into a cross-hair) then draw a bitmap container in your window (the size is irrelevant).
2. Select the bitmap container:



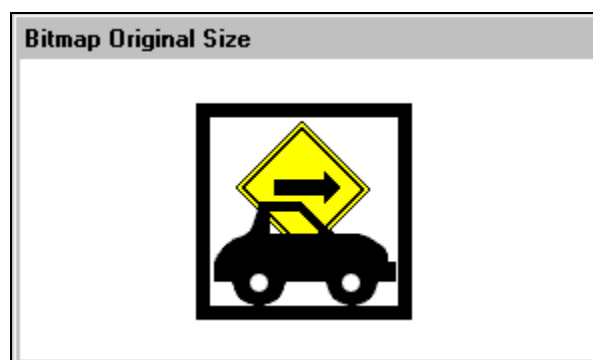
3. On the **Edit** menu, click **Import Image**. The Windows **Select Image File** dialog box appears:
  - ☞ To quickly paste the image, right-click the bitmap container, and then click **Import Image**.




4. Locate and select the .BMP, .PCX, .TGA or .JPG file that you want to import as a bitmap, then click **Open** or double-click the image filename. The image will be pasted into your bitmap container:



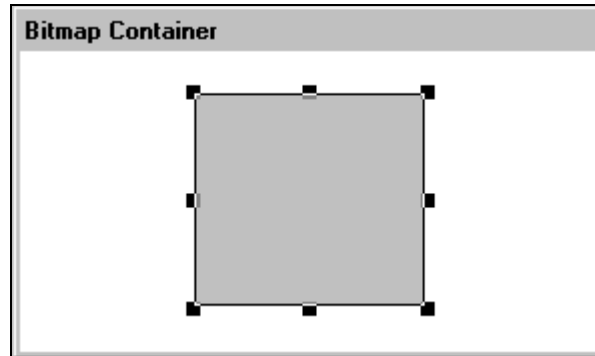
5. To make the bitmap its original size, select it, then on the **Edit** menu, click **Bitmap - Original Size**. The bitmap will be redrawn at its original size.
  - ☞ To quickly size the bitmap, right-click the bitmap, and then click **Bitmap - Original Size**.



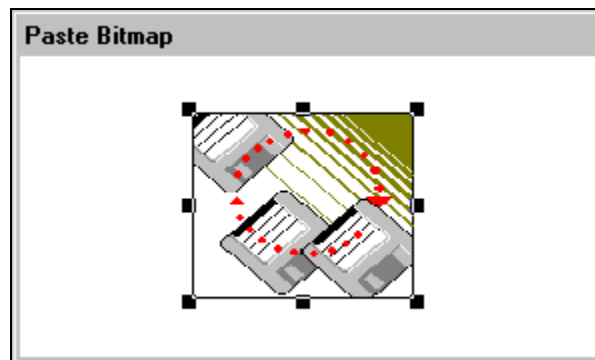
## Pasting a Bitmap from the Windows Clipboard

-  **To paste a bitmap from the Windows Clipboard into a window:**

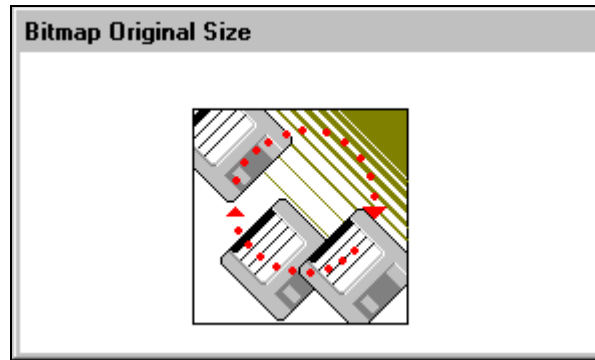
1. Copy the graphic to the Windows Clipboard. For example, display the graphic, then hold down the ALT key while you press the PRINT SCRN key to copy it to the Windows Clipboard.
2. Click the Bitmap tool (your cursor turns into a cross-hair), then draw a bitmap container in your window (the size is irrelevant).
3. Select the bitmap container:



4. On the **Edit** menu, click **Paste Bitmap**. The bitmap from the Windows Clipboard will be pasted into the bitmap container:
  - ☞ To quickly paste the bitmap, right-click the bitmap container, and then click **Paste Bitmap**.




5. To make the bitmap its original size, select it, and then on the **Edit** menu, click **Bitmap - Original Size**. The bitmap will be redrawn at its original size:
  - ☞ To quickly size the bitmap, right-click the bitmap, and then click **Bitmap - Original Size**.

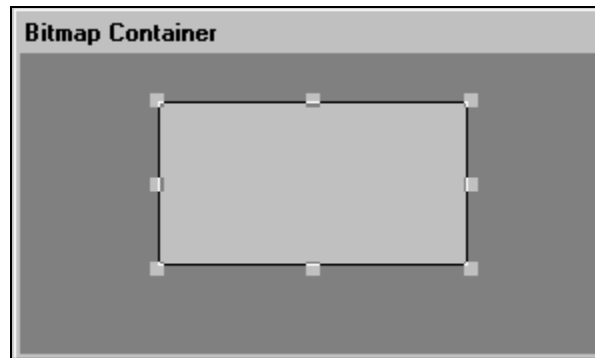


## Creating a Transparent Bitmap

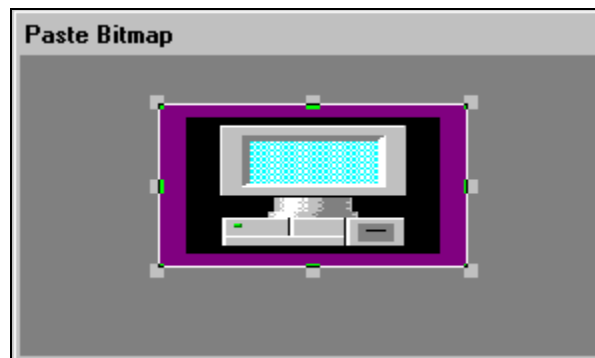
You can define a bitmap or image with a transparent color, so that you can float it over other objects. When you define a bitmap or image with a transparent color, the window's background or any objects behind the bitmap will show through it everywhere the transparent color is used.

➤  **To create a transparent bitmap:**


1. Click the Bitmap tool (your cursor turns into a cross-hair) then draw a bitmap container in your window (the size is irrelevant).
2. Select the bitmap container:



3. Right-click the bitmap container, and then click **Paste Bitmap** (if you have copied the graphic to the Windows Clipboard) otherwise, click **Import Image** (to locate and select the .BMP, .PCX, .TGA or .JPG file to open). The bitmap image will be pasted into the bitmap container:

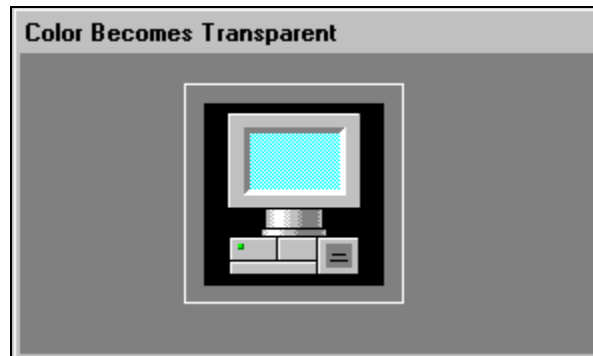





4. Right-click the bitmap, and then click **Bitmap - Original Size** to return the bitmap to its original size.
5. With the bitmap selected, click the Transparent Color tool  on the **Format** toolbar to open the transparent color palette.
6. Right-click a blank color square in the **Custom Palette** section at the bottom of the color palette. The **Edit Custom Color** dialog box appears:



7. Click the Blotter tool (the **Edit Custom Color** dialog box will close).
8. Click the color in the bitmap that you want to make transparent. The color will be copied to the color square that you selected in the transparent color palette.
9. Click the color square to apply the transparent color to the bitmap:



-  In this example, we made the wide border area of the bitmap transparent. Therefore the window background color now shows through the bitmap. If objects were behind the now transparent area, they would also shown through the bitmap.

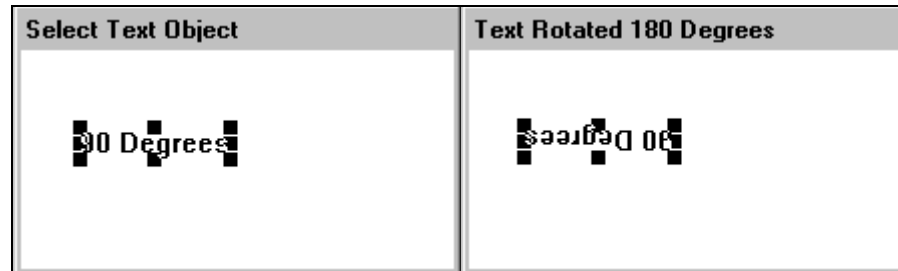
---

**Note** Only one transparent color can be applied per bitmap.

---

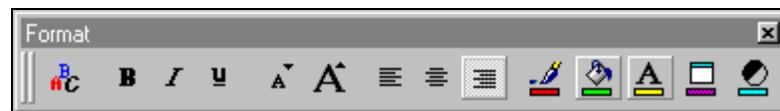
## Working with Text Objects

In WindowMaker you can change the font, font style, font size, justification and rotation of any selected text object. You can also rotate it 360 degrees by 90 degree increments (90 degrees, 180 degrees, 270 degrees and 360 degrees). For example:



**Note** Rotating text objects in WindowMaker has nothing to do with dynamically rotating objects in runtime. **Orientation** animation links cannot be applied to text objects. Therefore, text objects cannot be rotated in WindowViewer.

The **Format Toolbar** contains tools that you can use to quickly apply most of the commands found on the **Text** menu to selected objects. For example:



For more information on the **Format Toolbar**, see Chapter 1 - WindowMaker Program Elements.

## Formatting Text Objects

All WindowMaker text commands operate on single or multiple text string selections and numeric value fields. If no text object is selected when a command on the **Text** menu is executed, the command is automatically applied to the respective text tool's default setting on the **Format Toolbar** and the default setting of the Text tool on the **Draw Object Toolbar**.

The text justification attribute settings are particularly important for text objects used for outputting dynamic values. The justification determines how fields of varying length will be displayed in runtime.

For example, if you are displaying a numeric value at the end of a text string that is centered or is right justified, the entire text string, including the value will be centered again or justified again each time there is a change in the number of displayed digits.

For more information on the WindowMaker toolbars, see Chapter 1 - WindowMaker Program Elements.

## Displaying Numeric Values

Text objects are also used to display static or dynamic numeric values. By attaching a **Touch Links User Inputs - Analog** or **Value Display - Analog** animation link, to a text object you can display the value of an analog (integer or real) tagname.

To determine the display format of the analog value, the following four characters are used:

- 0** - zero
- #** - number or pound sign
- ,** - comma
- .** - period or decimal point

The following illustrates field formatting for analog values:

- #** Displays any whole number,  
e.g., **1234** would display as **1234**  
(Only 1 # sign is necessary)
- 0.0** Forces one leading zero and one decimal place;  
e.g., **.1** would display as **0.1**  
e.g., **77.1** would display as **77.1**
- 00000** Forces leading zeros as required;  
e.g., **123** would display as **00123**  
e.g., **1234** would display as **01234**  
e.g., **12345** would display as **12345**
- #,##0.0** Inserts comma and leading zero if required,  
and one decimal place;  
e.g., **1234.56** would be displayed as **1,234.6**  
e.g., **123.4** would display as **123.4**
- 0,000.0** Forces comma, leading zeros and one decimal place.  
e.g., **12.3** would display as **0,012.3**

---

**Note** If you use a zero in the format, it must be followed by zeros. All places to the right of the decimal point must always be zeros. For example, **000.00** is correct, while **#0#0.0#** is incorrect.

---

☞ All normal text formatting applies to numeric values. These include font, size, color justification and bolding.

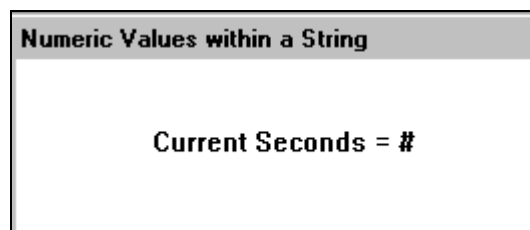
➤  **To create a text object:**

1. Click the Text tool in the **Draw Object Toolbar**.
2. Click in the window and type the text string.

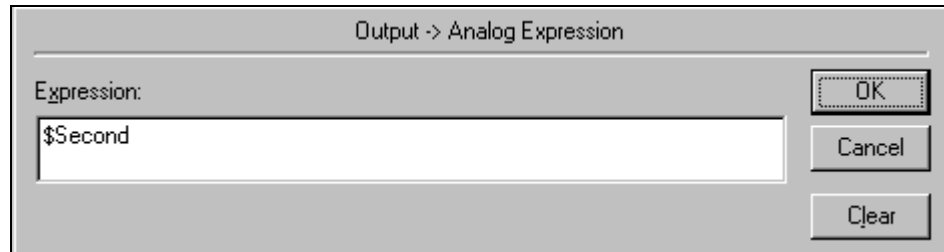
☞ To quickly access the various commands that can be applied to a text object, right-click the text object, and then click the appropriate command.


➤  **To display a numeric value within a text string:**

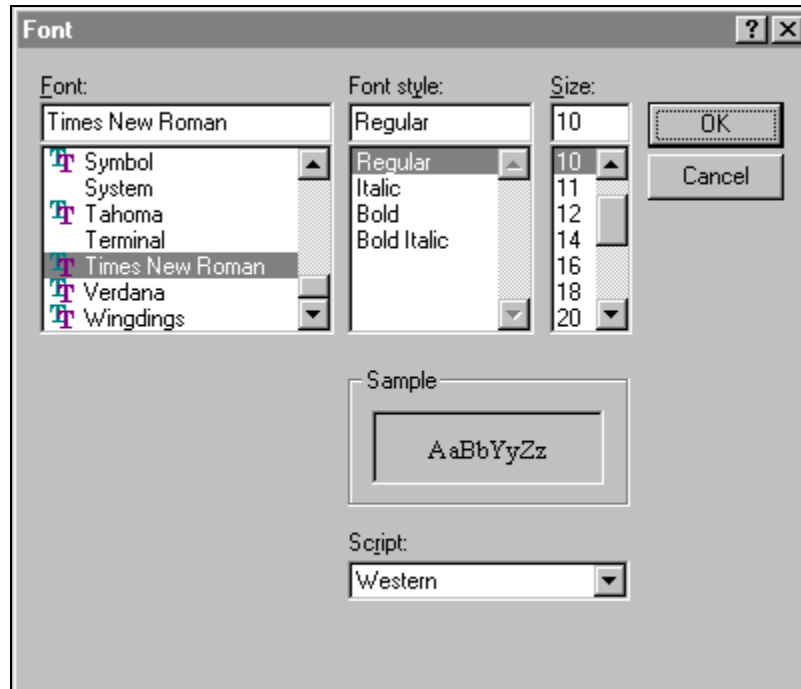
3. Click the Text tool, and then type a text object in the window using one of the previously described valid numeric formats. For example:



4. Select the object, and then on the **Special** menu, click **Animation Links** or double-click the text object. The animation links selection dialog box will appear.
  - ☞ To quickly access the dialog box, right-click the text object, and then click **Animation Links**.
5. In the **Value Display** section, click **Analog**. The **Output -> Analog Expression** dialog box appears:



6. In the **Expression** box, type an analog tagname or expression. (In this example, the system tagname **\$Second** is being used.)
  7. Click **OK**.
  8. Click the **Runtime** fast switch in the upper right hand corner of the menu bar (or use the short cut keys **ALT + !**) to switch to WindowViewer or, on the **File** menu, click **WindowViewer**.
  9. If you used this example, you will see the current system seconds (a value between 0-59) displayed in place of the pound (#) sign in the text string.
  10. Click the **Development** fast switch in the upper right hand corner of the menu bar (or use the short cut keys **ALT + !**) to return to WindowMaker or, on the **File** menu, click **WindowMaker**.
-  **To change the font, font style and font size of a string:**
1. Select the text string, and then on the **Text** menu, click **Fonts**, click the Fonts tool on the **Format Toolbar**. The standard Windows **Font** dialog box appears:



2. Select the desired font from the **Font** list (the font name will appear in the **Font** field). Once a font is selected, the styles and sizes available for it will appear in the **Font Style** and **Size** fields. When a font size is selected a sample of the font in the selected style and size will appear in the **Sample** field (see above example).
3. Click **OK**.

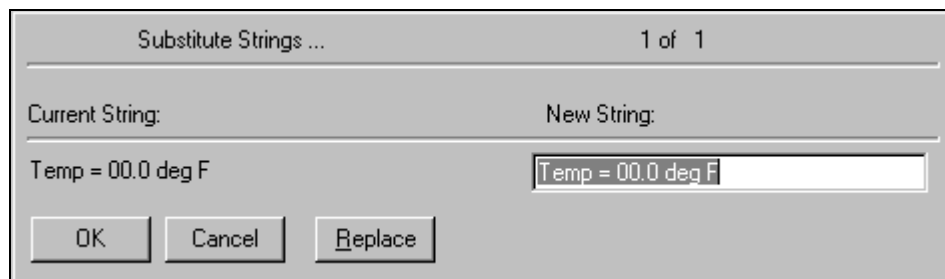
---

**Note** A font's point size will be enlarged or reduced in accordance with the range of point sizes available for the selected font. The default font for WindowMaker is System and cannot be sized. Choose a Windows True-Type font before changing the size.

---

## Editing Text Objects

- **To change the text in an object:**
  1. Select the object or button with the text.
  2. On the **Special** menu, click **Substitute Strings**. The **Substitute Strings** dialog box appears:
    - ☞ To quickly access the dialog box, right-click the text object, point to **Substitute**, and then click **Substitute Strings**.



☞ For more information on Analog Input/Display links, see Chapter 4 - Creating Animation Links.

3. In the **New String** box, type the new string, and then click **OK**.

☞ You can also use this command on strings that are included in a symbol or cell and to change the label on buttons drawn with the Button tool.

When you change a text string, it retains all of its original attributes, that is font, style, color, and so on. All normal text formatting also applies to numeric values.

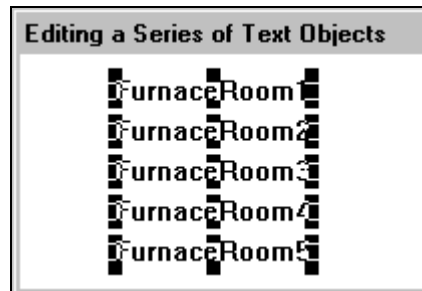
You can also select and edit multiple string objects at the same time.

## Replacing a Portion of a Text Object

You can change a portion of a text object's text and OpenHMI will automatically make the change to all selected text objects using the same text.

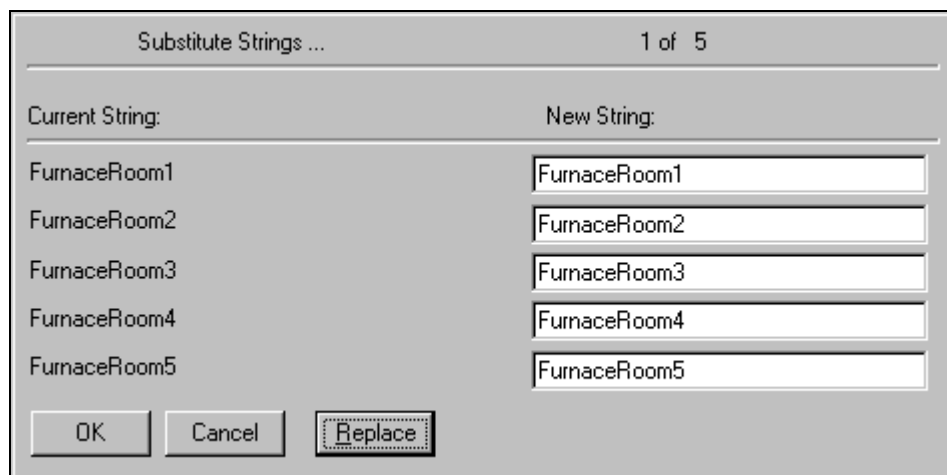
➤ **To change a portion of text in a series of text objects:**

1. Select all of the text objects.



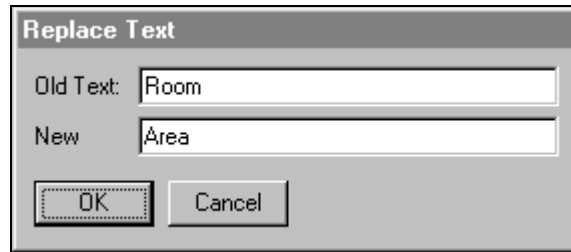
2. On the **Special** menu, click **Substitute Strings**. The **Substitute Strings** dialog box appears:

☞ To quickly access the dialog box, right-click a text object, point to **Substitute**, and then click **Substitute Strings**.

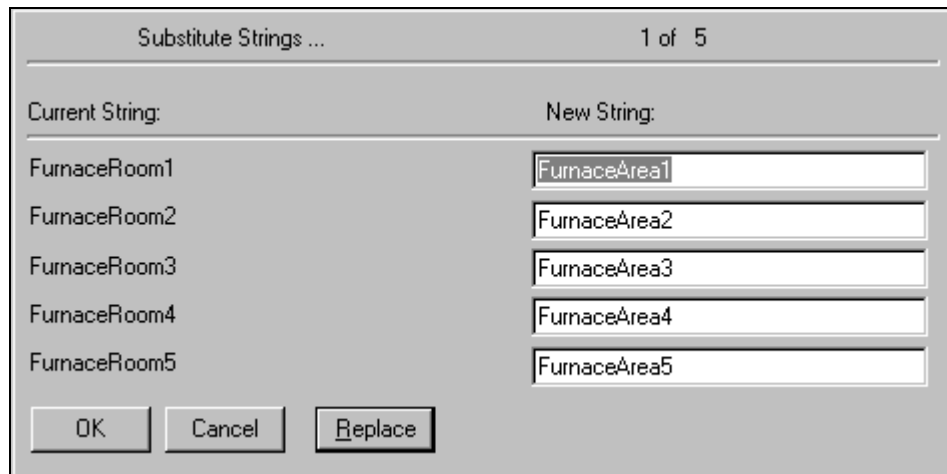


☞ If you right-click any of the text entry boxes in any dialog box, a menu will appear displaying the commands that you can apply to the selected text.

3. Click **Replace**. The **Replace Text** dialog box appears.



4. In the **Old Text** box, type the portion of the string that you want to replace.
5. In the **New** box, type the replacement text.
6. Click **OK**. The **Substitute Strings** dialog box reappears showing the change made to the selected text strings:

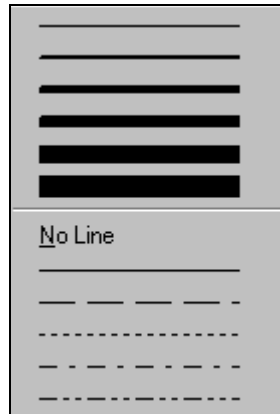


7. Click **OK**. All of the selected text objects will automatically be modified.

## Working with Lines and Outlines

You can change the style and width of a line object including the outlines around ellipses, rectangles, polygons and bitmaps or images. You can apply a line style or width change to a single selected object or multiple selected objects.

The **Line** menu is divided into two sections. The top section contains the line widths and the bottom section contains the line styles. For example:



➤ **To apply a line command:**

Select the object, and then on the **Line** menu, click the desired line style or width.

☞ If you do not select an object when you select a line style or width, the change will be applied to the default settings for all line tools in the **Wizard Toolbar**.

---

**Note** You can only change the width of solid lines. Broken lines can only be single pixel wide. Wider lines take longer to draw in runtime.

---

➤ **To remove an object's outline:**

Select the object, and then on the **Line** menu, click **No Line**. The object's outline will be removed.



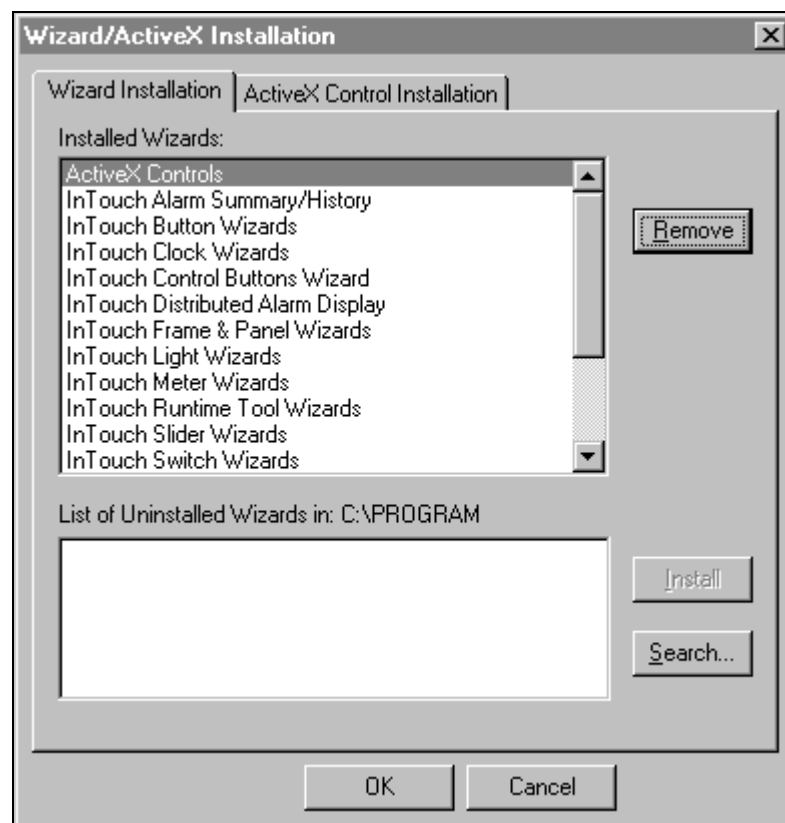
## Working with Wizards

Wizards save you a considerable amount of time during application development. They are easy to use and easy to configure. To configure a wizard, you install it, select it in the **Wizard Selection** dialog box, paste it into your window, and then double-click it. Its respective configuration dialog box will appear (assuming that it is a wizard that can be configured).

For example, if you wanted to use a slider wizard, you would need to configure items such as the tagname effect, the minimum and maximum range labels for the slider, the fill color, and so on. You can save a considerable amount of development time by using Wizards because you don't have to draw the individual components for the object, or set the value ranges for the object, or animate the object.

### To install or remove wizards:

1. On the **Special** menu, point to **Configure**, and then click **Wizard/ActiveX Installation**, or in the Application Explorer, double-click **Wizard/ActiveX Installation**. The **Wizard/ActiveX Installation** dialog box appears with the **Wizard Installation** property sheet active:
  - In the Application Explorer, you can also right-click **Wizard/ActiveX Installation**, and then click **Open**.



2. In the **Installed Wizards** list, select the wizard(s) that you want to remove from your application, and then click **Remove**. A message box will appear asking you to confirm the deletion.

---

**Note** The **Remove** button is active only when wizards are displayed in the **Installed Wizards** list.

---

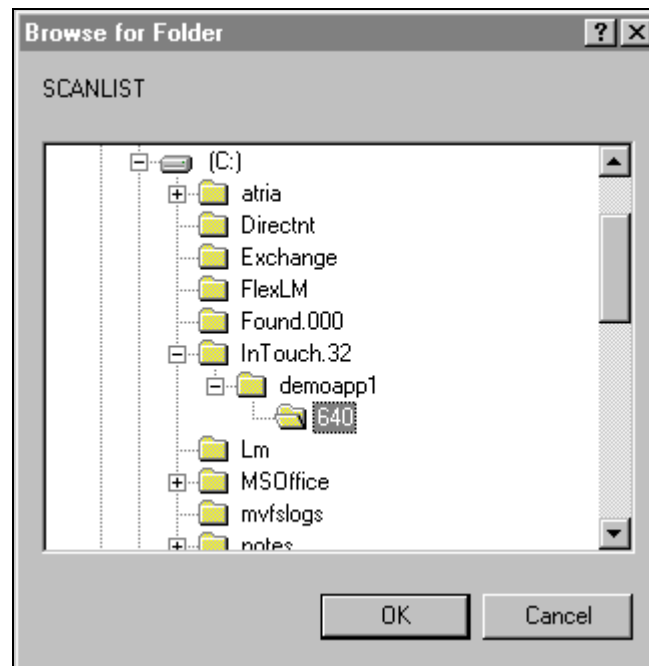
- ☞ To select a group of wizards, click the first wizard in the list, hold down the SHIFT key and click the last wizard that you want to select. All wizards in the list between your first and last selection will be selected. To select multiple wizards that are not consecutively listed, click the first wizard, hold down the CTRL key, and then click the next wizard. Repeat this for all wizards that you want to select.
3. Click **Yes** to remove the wizard. The removed wizard(s) is moved to the **List of Uninstalled Wizards** list.
    - ☞ When you remove a wizard, it is not deleted. However it is no longer loaded into memory.
  4. To install wizards, select them in the **List of Uninstalled Wizards**, and then click **Install**.

---


**Note** The **Install** button is active only when wizards are displayed in the **List of Uninstalled Wizards** list.

---

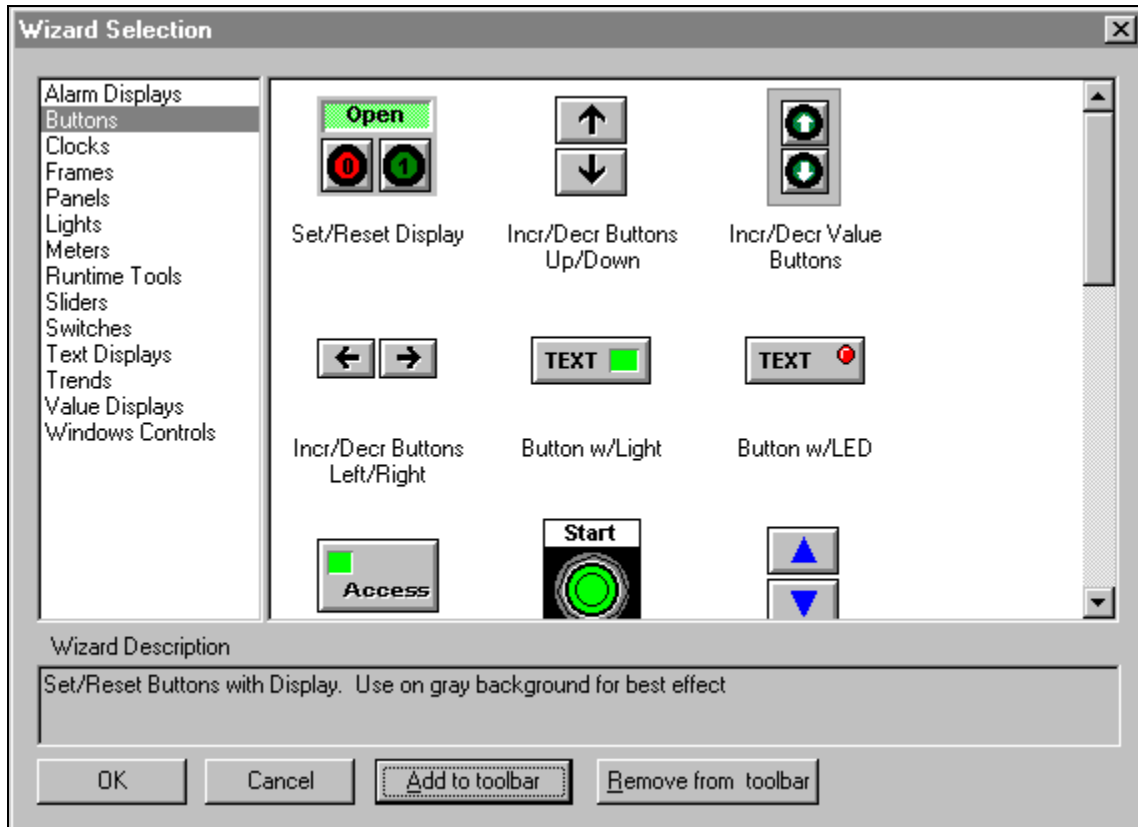
5. Click **Search** if you want to install wizards from another directory. The **Search for Wizard files** dialog box appears:



6. Locate the directory containing the wizards that you want to install, and then click **OK**. The wizard installation dialog box will reappear.
7. Any Wizards that were found will appear in the **List of Uninstalled Wizards** list and you can now install them as previously described.

➤  **To place a wizard in a window:**

1. Click the Wizard Dialog tool in the **Wizards/ActiveX Toolbar**. The **Wizard Selection** dialog box appears:




2. In the list of wizards, click the category of wizards that you want to use.
  - ☞ All available wizards in that category will be shown the display area. For example, if you select **Buttons**, all available button wizards will immediately be shown in the display area.
3. Select the wizard that you want to use, and then click **OK** or double-click the wizard. The dialog box will close and your window will reappear.
  - ☞ To add the wizard to the **Wizards/ActiveX Toolbar**, click **Add to toolbar**. Once you add a wizard to the **Wizards/ActiveX Toolbar**, you can select it and paste it into your open window at any time.

---

**Note** The number of wizards that you can add to the toolbar is limited to your system resources.

---


4. The cursor will change to a corner symbol  when you return to the window. Click the location in the window where you want to paste the wizard.
5. Double-click the wizard to configure it (if applicable).

---

**Note** Some toolbar functions may be used to modify applicable wizards directly. For example, the Reduce Font tool, Line Color tool, Fill Color tool, and so on.

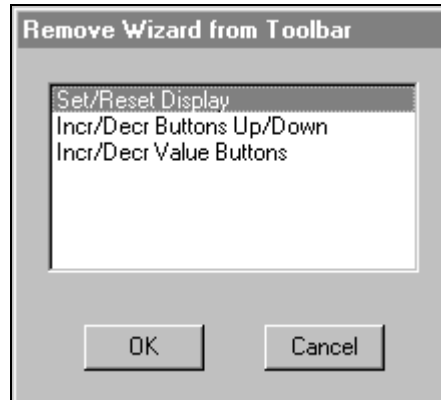
---

☞ For more information on the WindowMaker toolbars, see Chapter 1 - WindowMaker Program Elements.

➤  **To remove wizards from the toolbar:**

1. Click the Wizard Dialog tool in the **Wizards/ActiveX Toolbar**. The **Wizard Selection** dialog box will appear.

2. Click **Remove from toolbar**. The **Remove Wizard from Toolbar** dialog box appears:



3. Select the wizard(s) that you want to remove from the toolbar.
4. Click **OK**.

## OpenHMI Windows Control Wizards

The windows control wizards are complex objects. Unlike normal wizards, they provide enhanced functionality through OpenHMI QuickScripts. They can be used for editing data objects and operator inputs. Windows control wizards also have OpenHMI tagname .fields and some of their properties are accessible both during development and runtime when their QuickScript functions are used.

You can use windows control wizards in your OpenHMI application to display text/data, gather user input or offer choices for the user at runtime. Choices may be in the form of list boxes, check boxes, combo boxes and radio (option) buttons. You can use the text boxes to display or input text/data.

When you configure a windows control wizard, you must specify a **Control Name** to identify the control. OpenHMI uses the **Control Name** to identify the control when you execute a windows control QuickScript function. Therefore, you must also specify the **Control Name** parameter in the QuickScript function. For example:

```
SetPropertyD ( "ControlName.Property", Discrete );
```

**Control Names** do not add to the application's tagname count and must be unique for each control. Tagnames, although not required, are essential for productive use of the control. For example, selecting an item in a list box is not useful if the selected item is not automatically assigned to a tagname, thus making it accessible to OpenHMI.

Windows controls have properties (similar to tagname .fields) that can be modified at development (WindowMaker) and runtime (WindowViewer). They also support specific QuickScript functions that can be processed at runtime to modify lists, load files, disable controls, and so on.

---

**Note** To function properly, windows control objects cannot overlap each other. For verification, select the object in WindowMaker and insure that the object's handles do not touch another object.

The initial value of tagnames assigned to either a list box or combo box cannot be used to initialize the value of the list box or combo box.

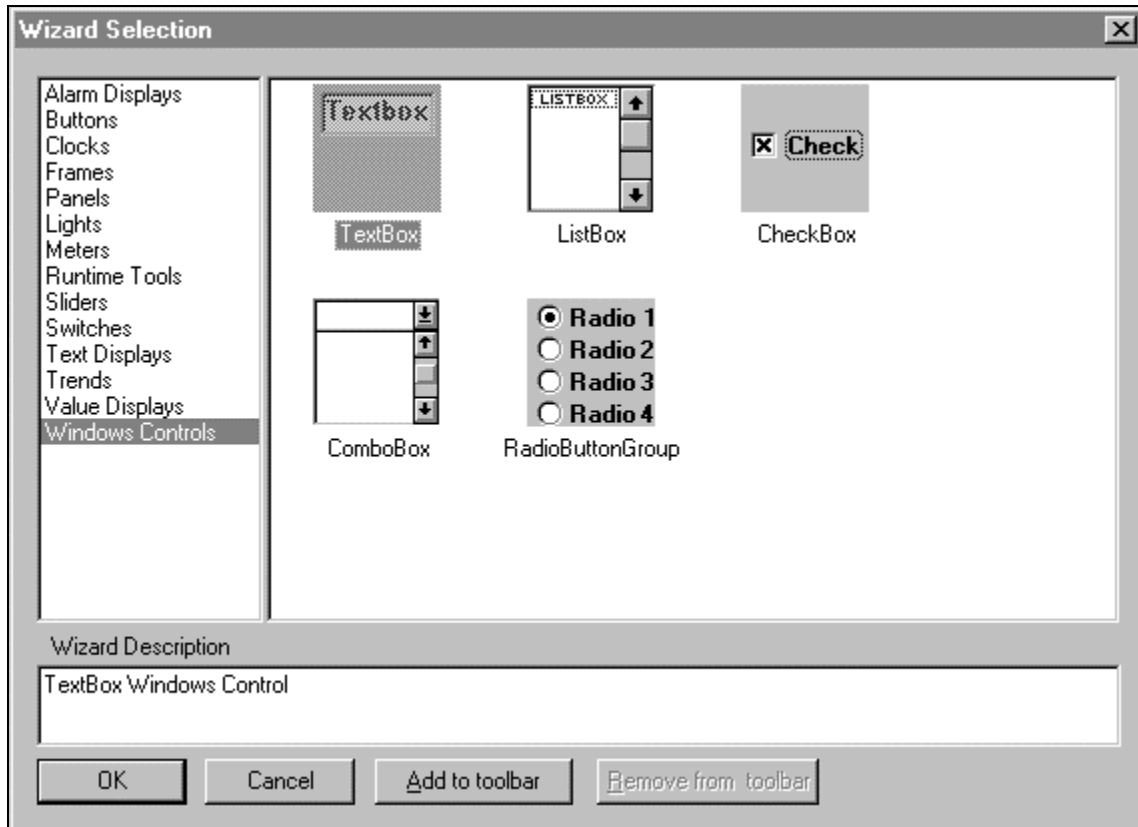
---

## Using OpenHMI Windows Control Wizards

The Windows control wizards available include: Text Boxes, Check boxes, Combo Boxes, List boxes and Radio (Option) Buttons. The windows control wizards also have tagname .fields and QuickScript functions that you can use to dynamically control them in runtime.

📄 Windows control wizards are pasted into your windows just like any other wizard.

🔗 For more information on pasting wizards, see ["To place a wizard in a window."](#)



- ☞ To achieve the best windows control 3-D effect, select a gray background for your window. If your window color cannot be gray, place a gray "Panel Wizard" behind the windows control wizard.

## Windows Control Usability Guidelines

It is very important that you follow the guidelines below when you are using windows control wizards:

1. Windows control wizards work properly when they do not overlap other windows control wizards or other normal graphic objects.
  - ☞ To verify that the windows control wizard is not overlapping any other object, select it in WindowMaker. Verify that none of its selection handles are touching another graphic object on the screen.
2. Windows control wizards should be used sparingly and intelligently.
  - ☞ Placing 10 to 20 windows control wizards in one window results in non-intuitive, hard to navigate displays. When it is necessary for you to use numerous windows control wizards, we recommend that you call other dialog boxes with additional windows control wizards.

## Text Box Control Wizard



Text boxes control wizards are versatile controls that can be used to get input from the user or to display text such as a notepad file (ASCII flat files only). You can configure text boxes to allow user input or as read-only for display purposes only. You can only assign **Message** type tagnames to a text box control wizard.

Text box control wizard QuickScript examples:

```
wcLoadText ("TextBox_1", FileName);
wcSaveText ("TextBox_1", FileName);
```

---

**Note** If the tagname has been defined with a maximum length, only that number of characters can be assigned from the text box contents to the tagname. If no tagname is assigned to the text box, its contents can be up to 65,535 characters.

---

## List Box Control Wizard



List box control wizards display a list of choices to the user. The choices are displayed vertically in a single column. If the number of items exceeds what can be displayed in the list box, scroll bars will automatically appear on the control. List boxes require the user to choose from a list and do not allow user input. You can only assign **Message** type tagnames to a list box control wizard.

List box control wizard QuickScript example:

```
IF (ItemToAdd == "") THEN
  Show "Cannot Add Blank";
ELSE
  wcAddItem ("ListBox_1", ItemToAdd);
  {Get the index of the item we just added.}
  {Since the list is sorted, we cannot assume anything about the new items
  location.}

  GetPropertyI ("ListBox_1.NewIndex", ListBox_NewIndex);
  {Now, set the Item Data specified on the screen by the user.}
  {From this point on, this item will have this data associated with it.}
  {It allows you to associate a number with a string; the string being displayed in
  the list.}


  wcSetItemData ("ListBox_1", ListBox_NewIndex, ListBox_ItemData)
  ;
  {Since we just added an item, update the "NumItems" variable.}
  GetPropertyI ("ListBox_1.ListCount", ListBox_NumItems);
ENDIF;
```

List box and combo box control wizards use an internal, one-based numbering system (item index) that automatically assigns a number to each item in the list. For example, the first item in the list is assigned the number 1, the second is number 2, and so on. The item index is a 32-bit integer that is used as a parameter for windows control "item"

---

**Note** When using list boxes and combo boxes with the **wcLoadList()** and **wcSaveList()**, specific formatting and information must be provided.

---

 For more information, on the windows control QuickScript functions, see your *OpenHMI Reference Guide*.

## Combo Box Control Wizard



Combo box control wizards combine the features of a text box and a list box. The choices are displayed vertically in a single column. If the number of items exceeds what can be displayed in the list box, scroll bars will automatically appear on the control. Combo box control wizards allow the user to make a selection, either by typing text or selecting an item from the list. You can only assign **Message** type tagnames to a combo box control wizard.

There are three styles of combo boxes:

Type	Description
------	-------------

---

<b>Simple</b>	Combo boxes display their list at all times. To display the entries in the list box, the list box must be drawn large enough to display all entries. A vertical scroll bar is automatically inserted if there are more entries than can be displayed. Simple combo boxes allow the user to type in choices that are not in the list or will display the first item in the list matching the typed letters. If no match is found, the top of the list is displayed.
<b>Drop Down</b>	Combo boxes allow the user to either type text directly or click an arrow to open a list of choices. As with the simple combo box, this control allows the user to type choices not on the list or will display the first item in the list matching the typed letters. If no match is found, the top of the list is displayed.
<b>Drop Down List</b>	Combo boxes are similar to simple list boxes. They display a list of choices to select. Unlike list boxes, however, the list is not displayed until the arrow is clicked. This type of control is used to conserve screen space.

Combo box control wizard QuickScript example:

```
wcAddItem("ComboBox_1", UserMessage );
```

Where: *UserMessage* is a tagname assigned to a string input link. When the operator types a new message, and then clicks this action pushbutton, linked to the "**On Down**" QuickScript, the message is displayed in the combo box wizard with the control name "ComboBox\_1."

## Check Box Control Wizard



A check box indicates whether a particular condition is on/off, true/false or yes/no. Check boxes work independently of each other, allowing the user to select or deselect any number of check boxes at the same time. Check boxes return a discrete value. They return 0 if not selected and 1 if selected. You can only assign **Discrete** type tagnames to a check box control wizard.

Check box control wizard QuickScript example:

```
{ Clear any previous machine }
Machine = "";

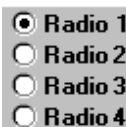
IF (Cutter_Selected) THEN
    Machine = Machine + "Cutter";
ENDIF;

IF (Mixer_Selected) THEN
    Machine = Machine + "Mixer";
ENDIF;
```

Where: *Cutter\_Selected* is the tagname assigned to the control name "Checkbox\_1" in the first check box wizard. *Mixer\_Selected* is the tagname assigned to the control name "Checkbox\_2" in the second check box wizard.

*Machine* is a tagname assigned to a string output link that displays the name for the check box selected.

## Radio (Option) Buttons Control Wizard



Radio, or option buttons present a set of choices for the user. Unlike check boxes, radio buttons operate as part of a group. Selecting one radio button immediately



clears all of the other buttons in the group. Radio buttons return an integer value. The value of a radio button control corresponds to the selected radio button.

For example, if Radio 1 option is selected, the current value is 1. If the Radio 4 option is selected, the value is 4. You can only assign **Integer** type tagnames to a radio button control wizard.

Radio (option) button control wizard QuickScript example:

```
SelectedMachine=1;
```

Where: *SelectedMachine* is an integer tagname assigned to the radio button wizard with the control name "RadioButtonGroup1. This is a Window **"On Show"** QuickScript that sets the value of the *SelectedMachine* tagname to 1. (This sets the default to select the first radio button in the group when the window is initially shown.) When the operator selects another radio button, the value of *SelectedMachine* will change according to the button selected. For example, if the radio button group had 4 choices and the operator selected the third button, *SelectedMachine's* value would be set to 3.

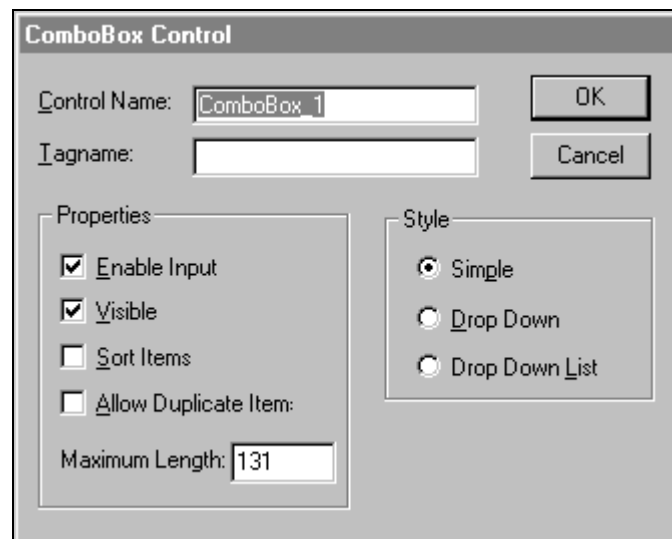
## Configuring a Windows Control Wizard

Each windows control wizard has its own unique configuration dialog box based on its intended functionality. The options shown in the dialog box are configurable properties not available through other WindowMaker tools. However, properties such as color, font style and size are modified by using the respective WindowMaker tools.

Most of the windows control wizards support QuickScript functions. For example, you could create a Data Change QuickScript to load and clear the lists, add and delete items in the lists, and so on.

### ➤ To configure a windows control wizard:

1. Paste the wizard in your window.
2. Double-click it. Its respective configuration dialog box will appear. For example:



- ☞ If you right-click any of the text entry boxes in any dialog box, a menu will appear displaying the commands that you can apply to the selected text.

3. In the **Control Name** box, type a unique name to identify the windows control.
  - ☞ You must specify a unique **Control Name** for each windows control wizard you use for correct operation at runtime. WindowViewer uses the **Control Name** to identify the control when you execute a windows control QuickScript function. **Control Names** must start with an alpha character (underscores and numbers can be used after the first alpha character) and cannot include any special characters.
4. Specifying a **Tagname** when you configure the windows control is optional. However, if you do specify a tagname, its value is automatically set to the **.Value** property of the control (that is, the item index for the item selected in a list box).
5. Type in all other required entries and set all parameters for the particular windows control that you are configuring, as applicable.
6. Click **OK**.

## Windows Control Wizard Properties

Windows control wizards have properties like OpenHMI tagname **.fields**. They can be read-write or read-only. Some properties are accessible at development and some at runtime. They are identified as *ControlName.x*, where *x* is the property.

For example, if the **.Visible** property of a windows control is equal to 0, the control will not be visible in the window. Similar to OpenHMI tagnames, **.Value** is the default property for the windows control wizard.

In WindowMaker, windows control wizard properties such as text font, size and color are modified using the respective WindowMaker toolbars or menu commands. The properties that are not supported by the toolbar or menu commands, are configured from within the wizard's configuration dialog box. Other properties of windows control wizards are dynamic and are read-write or read-only in runtime. This is similar to runtime properties of OpenHMI tagnames (**.fields**) such as **.Value**. Unlike OpenHMI tagnames, runtime properties for windows control wizards are accessed through QuickScript functions not animation link expressions.

The runtime properties may be either read-write or read-only depending on the property. The **GetProperty()** and **SetProperty()** QuickScript functions must be used to control or retrieve these properties. The following briefly describes each windows control property:

Property	Description
<b>.Caption</b>	Determines the "message" to be displayed with the check box.
<b>.Enabled</b>	Determines whether the control object can respond to operator-generated events.
<b>.ListCount</b>	Determines the number of items in a list box or combo box.
<b>.ListIndex</b>	Determines the corresponding index ( <i>tagname</i> or <i>number</i> ) of the currently selected item in the list.

---

**Note** Index is a number that defines a specific item in a list. When using a list box, an index of -1 indicates that no item is currently selected. When using a combo box, an index of -1 indicates that the user has entered new text into the text entry field of the control.

---

<b>.NewIndex</b>	Returns the corresponding integer index ( <i>tagname</i> ) of the last item added to the list box or combo box through the <b>wcAddItem()</b> or <b>wcInsertItem()</b> functions.
<b>.ReadOnly</b>	Determines whether the contents of the text box are read-only or read-write.
<b>.TopIndex</b>	Determines the corresponding integer index of the top most item in the list box.
<b>.Value</b>	The default property for all OpenHMI windows control wizards. Changes made to this property are synchronized in the OpenHMI tagname and the windows control wizards.
<b>.Visible</b>	Determines whether the windows control is visible in the window.

---

**Note** The windows control wizard properties do not appear in the **Choose field name** dialog box.

---

For example:

```
[ErrorNumber=]GetPropertyM("ControlName.Property", Tagname);
```

Where:

Parameter	Description
ControlName	The <b>Control Name</b> configured for the windows control wizard, for example, CheckBox_1 or the name of an alarm object, for example, AlmObj_1.
.Property	Windows control or alarm object property.
Tagname	A valid OpenHMI tagname (of the same type to be returned) that will hold the property value when the function is processed.

☞ For more information on windows control wizards, see "[OpenHMI Windows Control Wizards](#)."

## Windows Control Wizard Functions

The following briefly describes the OpenHMI QuickScript functions that are available for use with the OpenHMI Windows Control wizards:

Function	Description
<b>wcAddItem</b>	Adds the supplied string to the list box or combo box.
<b>wcClear</b>	Removes all items from the list box or combo box.
<b>wcDeleteItem</b>	Deletes the item associated with the item index argument in both list or combo boxes.
<b>wcDeleteSelection</b>	Deletes the currently selected item from the list. Applies to list boxes and combo boxes
<b>wcErrorMessage</b>	Given an error number, <b>wcErrorMessage()</b> , returns a string message describing the error. Applies to list boxes, text boxes, combo boxes, radio buttons and check boxes.
<b>wcFindItem</b>	Determines the corresponding index of the first item in the list box or combo box that matches the supplied string.
<b>wcGetItem</b>	Returns the value property of an item string associated with a corresponding index in a list box or combo box.

---

<b>wcGetItemData</b>	Retrieves the integer value associated with a list item in a list box or combo boxes.
<b>wcInsertItem</b>	Inserts a string into a list box or combo box.
<b>wcLoadlist</b>	Replaces the contents of the list box or combo box with new items.
<b>wcLoadText</b>	Replaces the contents of the text box with a new string.
<b>wcSavelist</b>	Replaces the contents of a filename with the items in a list object.
<b>wcSaveText</b>	Saves the text contained in a text box to a filename.
<b>wcSetItemData</b>	Assigns an integer value to an item in a list box.

## Working with ActiveX Controls

ActiveX controls, originally known as OLE controls or OCXs, are standalone software components that perform specific functions in a standard way. ActiveX controls define standard interfaces for reusable components. ActiveX controls are not separate applications. Instead, they are servers that are placed into a control container. To use ActiveX controls, they must be placed in an ActiveX container. OpenHMI is an ActiveX container. Microsoft VisualBasic and internet browsers are also ActiveX containers.

ActiveX controls behave exactly like OpenHMI Wizards, except they bring compelling new functionality to OpenHMI applications. You can create ActiveX controls by using Visual Basic, Microsoft VC++ or other 3rd party development tools. You can also buy ActiveX controls from third-parties for specific functionality. These controls are packaged in the OCX form.

There are three main components of ActiveX controls: *properties*, *methods* and *events*. Properties are very similar to variables that you can modify, for example, `Calendar.day`, `Control.height`, and so on. Methods are similar to script function calls that you can call from the container.

For example, `Browser.Navigate("http://www.xycomautomation.com")`, `Engine.start()`. Events occur through the ActiveX container. For example, `Control.click (shift)`, `FileViewer.DoubleClick (name)`, and so on.

OpenHMI allows you to access ActiveX control properties, methods and events. You can associate these properties with OpenHMI tagnames or you can access them through OpenHMI QuickScripting.

---

**Note** In order for an ActiveX Event script to function properly, the ActiveX control for which the script was created, must be loaded into memory. If the window containing an ActiveX control is closed, its ActiveX Event scripts, or any other OpenHMI QuickScripts containing script functions associated with that ActiveX control, will not execute properly.

---

You can use one or more ActiveX controls in your OpenHMI application. OpenHMI allows you to easily select and paste an ActiveX control into any application window and to add them to your **Wizards/ActiveX Toolbar**. You can also import ActiveX Event scripts from one application to another.

➤ **To use an ActiveX control in OpenHMI:**

1. Install the ActiveX control(s) that you want to use.
2. Select and paste the ActiveX control into a WindowMaker window.
3. Configure the ActiveX control's properties and assign them to tagnames.
4. Associate ActiveX events to ActiveX Event scripts.
5. Call ActiveX methods and set ActiveX control properties in ActiveX Event scripts, or other OpenHMI QuickScripts.

The following WindowMaker edits can be made to an ActiveX control:

- An ActiveX control's size can be changed, if sizing is supported by the control.
- ActiveX controls can be duplicated, cut, copied, pasted and deleted.
- All aligning commands (left, right, top, bottom, centerpoint) can be applied to an ActiveX control.
- ActiveX controls can be added to the **Wizards/ActiveX Toolbar**.
- ActiveX controls can be included with other objects when creating a cell.

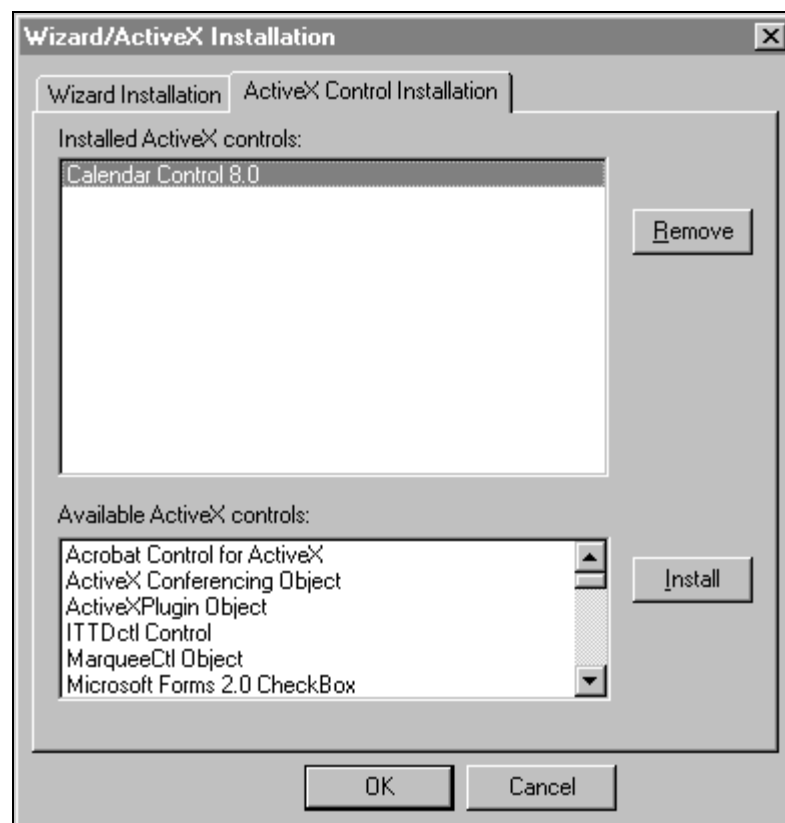
- The WindowMaker menu commands and their equivalent toolbar tools can be used to directly modify many ActiveX properties. For example, Reduce Font, Line Color, Fill Color, and so on.

OpenHMI does not support the following types of ActiveX controls:

- Windowless Controls
- Simple Frame Site (Group Box)
- Containers
- Data Controls
- Dispatch Objects
- Arrays, Blobs, Objects, Variant Types

➤ **To install or remove an ActiveX control:**

1. On the **Special** menu, point to **Configure**, and then click **Wizard/ActiveX Installation**, or in the Application Explorer, double-click **Wizard/ActiveX Installation**. The **Wizard/ActiveX Installation** dialog box appears.
  - ☞ In the Application Explorer, you can also right-click **Wizard/ActiveX Installation**, and then click **Open**.
2. Click the **ActiveX Control Installation** tab to activate the **ActiveX Installation** property sheet:



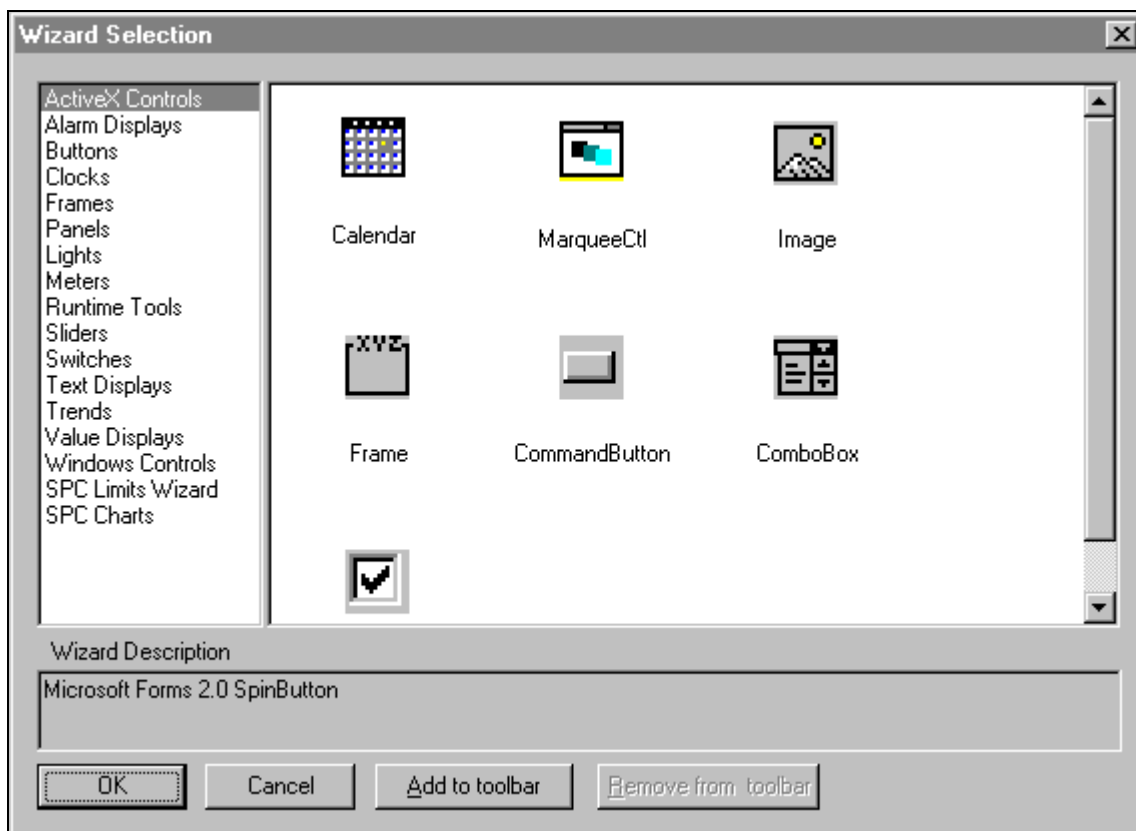
3. In the **Installed ActiveX controls** list, select the control(s) that you want to remove from your application, and then click **Remove**. An interactive message box will appear asking you to confirm the deletion.
  - ☞ To select a group of controls, click your first selection, hold down the SHIFT key and select your last selection. All controls in between will be selected

as well. To select multiple controls that are not consecutively listed, click the first control, and then hold down the CTRL key as you click another.

4. Click **Yes** to remove the control(s). The removed control(s) is moved to the **Available ActiveX controls** list.
  - ☞ When you remove a control, it is not deleted. However it is no longer loaded into memory. Therefore, it will not function properly.
5. To install ActiveX controls, select them in the **Available ActiveX controls** list, and then click **Install**.
  - ☞ The **Install** button is active only when controls are displayed in **Available ActiveX controls** list.
6. Click **Close**.

➤  **To place an ActiveX control in a window:**

1. Click the Wizard Dialog tool in the **Wizards/ActiveX Toolbar**. The **Wizard Selection** dialog box appears:





2. In the list of wizards, click the **ActiveX Controls** category. All available ActiveX controls will be shown the display area.
3. Select the ActiveX control that you want to use, and then click **OK**, or double-click the control. The dialog box will close and your window will reappear.
  - ☞ To add the ActiveX control to the **Wizards/ActiveX Toolbar**, click **Add to toolbar**. Once you add a control to the **Wizards/ActiveX Toolbar**, you can select it and paste it into your open window at any time.

---

**Note** The number of ActiveX controls that you can add to the toolbar is limited to your system resources.

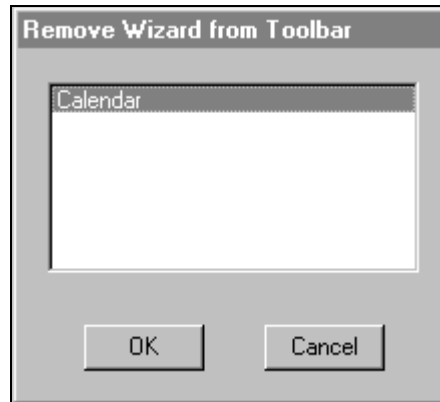
---

4. The cursor will change to the corner symbol, , when you return to the window. Click the location in the window where you want to paste the ActiveX control.
5. Double-click the control to configure its properties.

 For more information on the WindowMaker toolbars, see Chapter 1 - WindowMaker Program Elements.

➤  **To remove ActiveX controls from the toolbar:**

1. Click the Wizard Dialog tool in the **Wizards/ActiveX Toolbar**. The **Wizard Selection** dialog box will appear.
2. Click **Remove from toolbar**. The **Remove Wizard from Toolbar** dialog box appears:



3. Select the ActiveX control(s) that you want to remove from the toolbar.
4. Click **OK**.



## Configuring an ActiveX Control

When you paste an ActiveX control into an OpenHMI window, you must configure its properties to interact with OpenHMI. Each control must be named for reference from OpenHMI QuickScripts. A default control name such as, Calendar1, will be generated when you paste the ActiveX control. (This control name will be global within your OpenHMI application.)

The ActiveX control's properties must be assigned to OpenHMI tagnames. Each property type must be assigned to an equivalent OpenHMI tagname type.

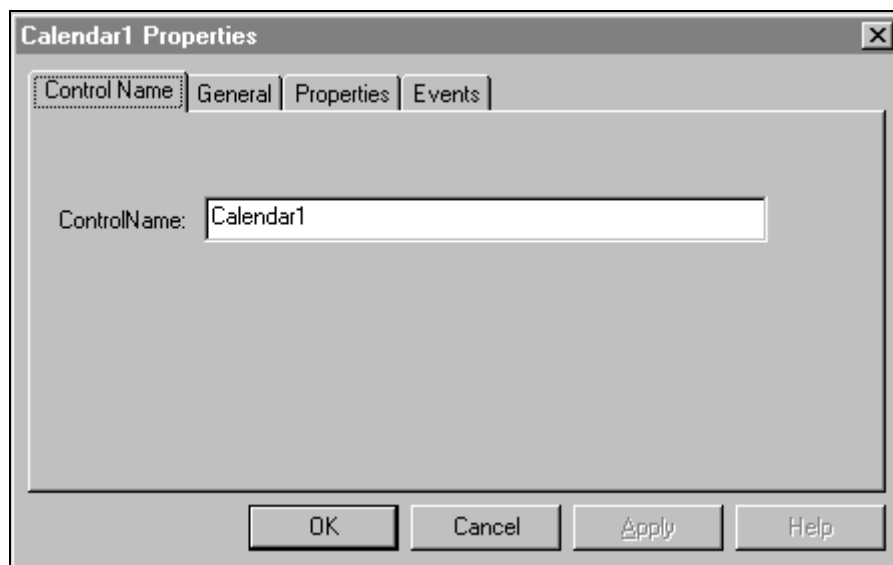
➤ **To name an ActiveX control:**

1. Paste the ActiveX control into your WindowMaker window.
2. Double-click the control, or right-click the control, and then click **Properties**. The control's respective **Properties** dialog box will appear.

---

**Note** Each ActiveX control's **Properties** dialog box is unique to the control. The number of tabs displayed is based upon the properties of the particular control. Some ActiveX controls may require you to configure more properties than others do. For example some controls may require you to configure their **Colors** and **Fonts**, while others may not have these properties. However, for all ActiveX controls, OpenHMI adds three tabs; **Control Name**, **Properties** and **Events**. For example:

---



3. Click the **Control Name** tab, and then type a unique name for the ActiveX control in the **ControlName** box.

☞ You must define a unique name for each ActiveX control used in your OpenHMI application. The Control Name is used in script functions to identify the control. For example:

```
#Calendar1.day = Tag1;
#Calendar1.year = 1997;
```

---

**Note** If you use the default Control Name, for example, Calendar1, and you subsequently duplicate the ActiveX control, OpenHMI will automatically

increment the Control Name. In this case, the duplicate ActiveX control's name would be Calendar2.

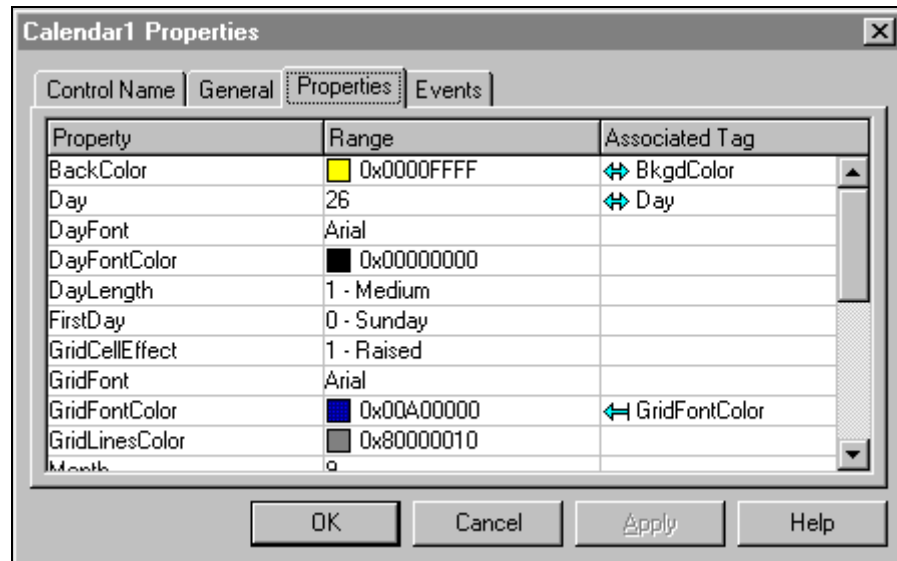
## Configuring ActiveX Control Properties

The properties that you can configure for a particular ActiveX control are determined by the ActiveX control designer. Each ActiveX control's **Properties** property sheet displays three columns: **Property**, **Range** and **Associated Tag**. The **Property** and **Range** columns are read-only. The **Associated Tag** column is used to associate OpenHMI tagnames with the respective property in the **Property** column.


**Note** When you click certain items in the **Range** column an arrow will appear that you can click to view the list of possible values for the item. The items in the list are for viewing purposes only and cannot be changed.

### ➤ To configure an ActiveX control's properties:

1. Click the **Properties** tab in the ActiveX control's **Properties** dialog box to activate the **Properties** property sheet:




2. Click in the middle of each cell in the **Associated Tag** column, and then type a tagname for the respective property.
  - ☞ If you type in a tag name that is not defined in the Tagname Dictionary, you will be prompted to define it now.

If you double-click a blank cell, or click the  button, the Tag Browser will appear displaying the tagnames for the selected tag source. Double-click the tagname that you want to use, or select it, and then click **OK**. The tagname is automatically inserted into the cell.









☞ For more information on the Tag Browser, see Chapter 3 - Tagname Dictionary.

3. Once you specify the tagname, double-click in the cell to the left of the tagname to select the association direction for the tagname to its respective property. (Continuously double-clicking will cycle you through the various association direction choices. The association direction choices are described below.)

- There are actually two fields in each cell in the **Associated Tag** column. The association direction selection and the tagname entry. The ActiveX control determines the association direction and the property type determines the tagname type that must be used.

You can select one directional or bi-directional association. However, if the association direction you select is not valid for the property or tagname, the control will automatically change it accordingly. For example, if you select , when the tagname's value changes, its associated property is changed accordingly. A certain subset of the symbols below will appear based upon the potential relationship between the property and the tagname.

For example, if you associate a property to a writeable tagname, you will see a different set of associations than if you associate the same property to a read-only tagname. Select the appropriate association symbol as follows:

-  The tagname sets the value of the associated property.
-  This symbol indicates that the property is read-only and the tagname cannot change the property's value.
-  The property sets the value of the associated tagname.
-  This symbol indicates that the tagname is read-only and the property cannot change the tagname's value.
-  Value can be set from both the tagname or the property. (Tagname takes precedence.)
-  The tagname and the property are both read-only.
-  The tagname can change the property's value, but the property cannot change the tagname's value. The property cannot change the tagname's value because the property is non-bindable, or the tagname is read-only.
-  The property can change the tagname's value, but the tagname cannot change the property's value. The tagname cannot change the property's value because the property is read-only.

#### 4. Click **OK**.

---

**Note** You can also access or change properties through ActiveX Event scripts and/or other OpenHMI QuickScripts. All ActiveX script functions are qualified by the # (pound) sign. The valid syntax to access ActiveX properties is:


```
#ControlName.PropertyName
```

Examples:

```
#Calendar1.Day = 29;
```

```
Tag1 = #Calendar1.year;
```

---

 For more information, see, "[Configuring an ActiveX Control.](#)"

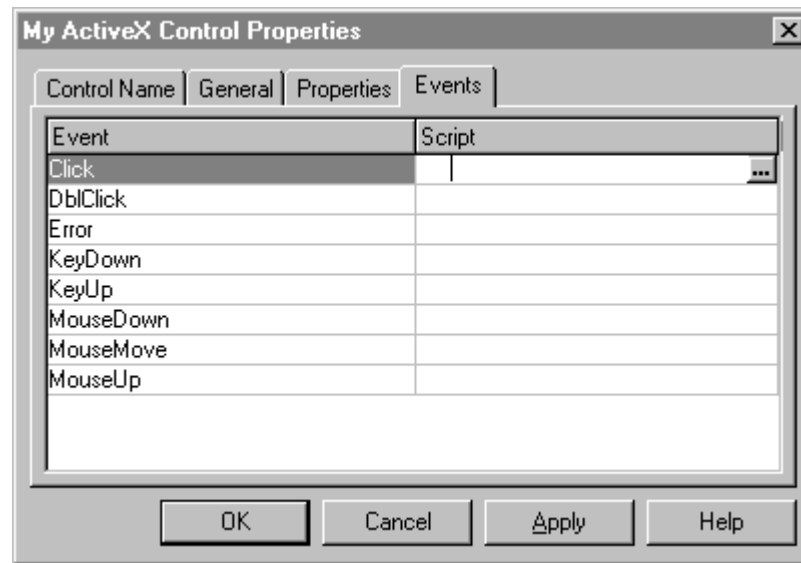
## Using ActiveX Control Methods

ActiveX control methods are similar to ActiveX control properties. You activate methods in runtime (WindowViewer). ActiveX control methods are accessed through ActiveX Event scripts and/or other OpenHMI QuickScripts.

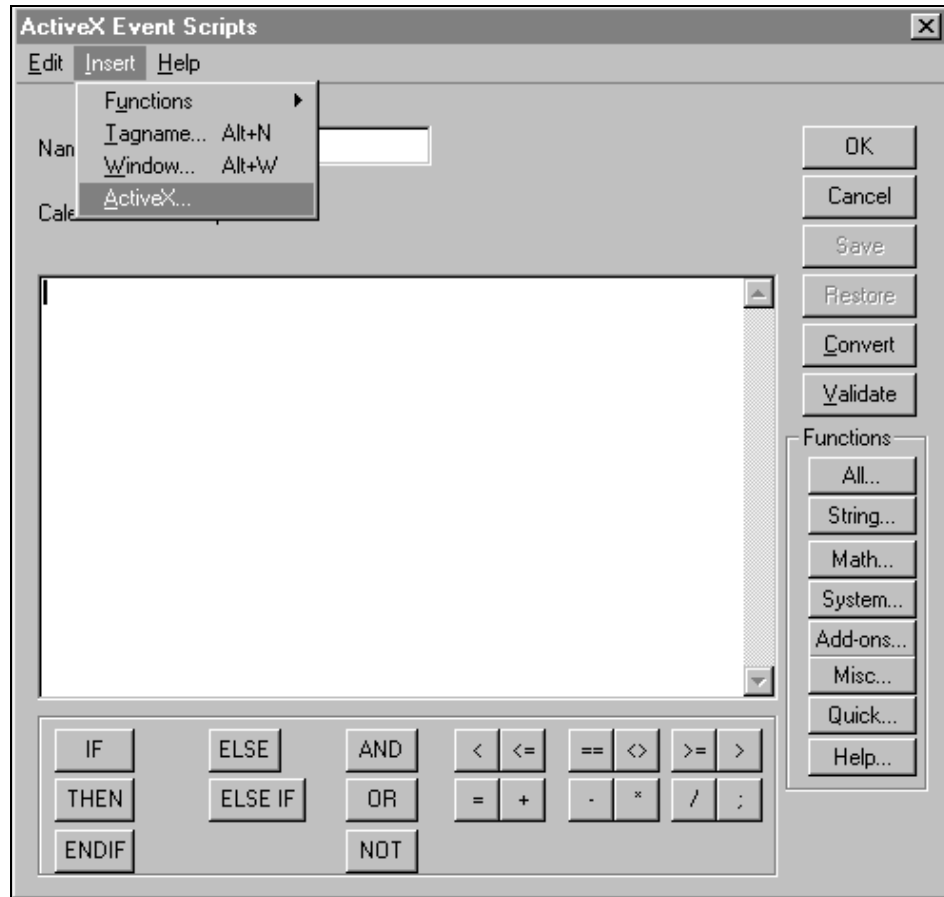
**Note** In order for an ActiveX Event script to function properly the ActiveX control for which the script was created, must be loaded into memory. If the window containing an ActiveX control is closed, its ActiveX Event scripts, or any other OpenHMI QuickScripts containing script functions associated with that ActiveX control, will not execute properly.

➤ **To use ActiveX methods and/or properties:**

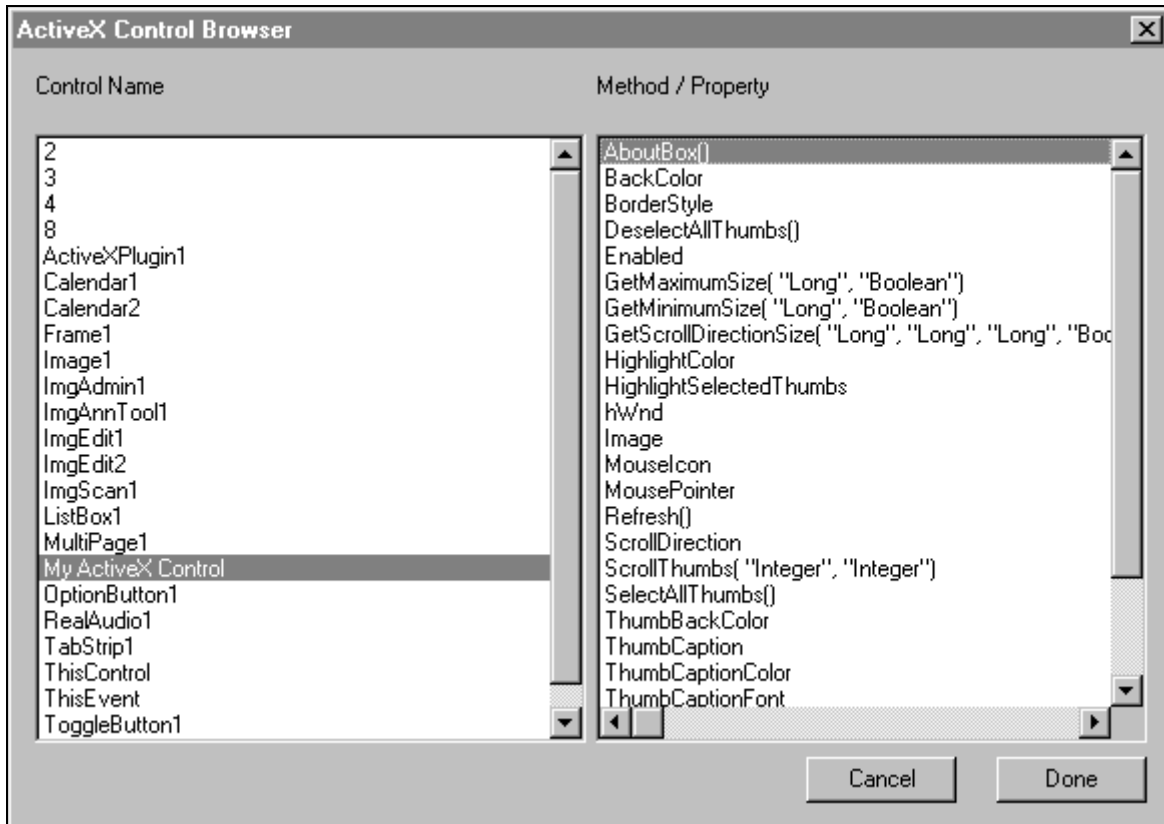
1. In the ActiveX control's **Properties** dialog box, click the **Events** tab to activate the **Events** property sheet:



2. Double-click a blank cell in the **Script** column. The **ActiveX Event Scripts** editor appears:



3. On the **Insert** menu, click **ActiveX**. The **ActiveX Control Browser** appears:



4. In the **Control Name** list, select the ActiveX control whose methods or properties you want to access.
  - ☞ The names of all ActiveX controls currently being used in your application will be listed.

---

**Note** If you select **This Control** instead of the actual **Control Name**, the methods and properties displayed will be those for the ActiveX control currently selected in your application. By selecting **This Control** instead of the actual **Control Name**, you can create generic ActiveX Event script functions. You can then copy and paste these functions into any other ActiveX Event script, or any other OpenHMI QuickScript without having to change the **Control Name** in the new script. For example:

```
#ThisControl.Navigate ("http:\\www.xycomautomation.com");
#ThisControl.Navigate(URL); { where URL is a tagname}
```

**This Control** is accessible only through ActiveX Event scripts. It is not accessible through any other type of OpenHMI QuickScript.

---

5. In the **Method / Property** list, select the method or property that you want to use in your script.
  - ☞ Properties are the items in the list that include parenthesis. For example, **Display()**.
6. Click **Done**. The selected control name and method or property are automatically inserted into your script.
  - ☞ ActiveX control's methods and properties are also accessed through the **Insert** menu in all other OpenHMI QuickScripts types.

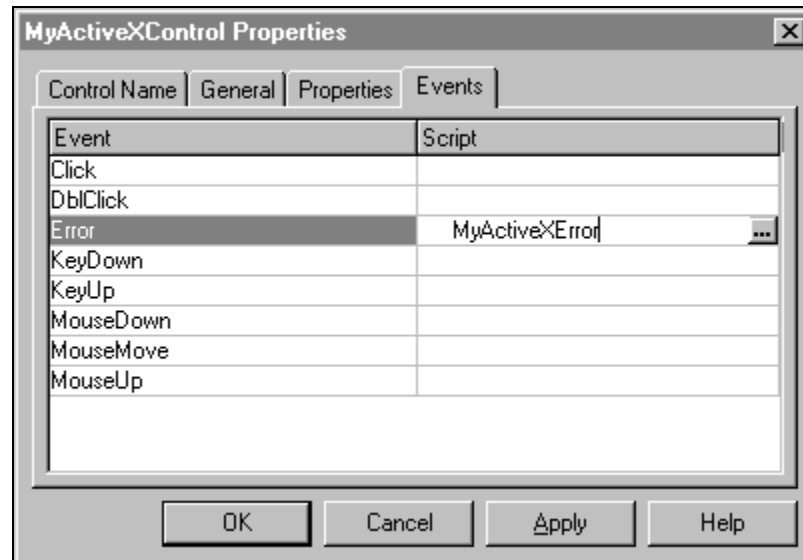
## Using ActiveX Control Event Parameters

You can execute ActiveX control events in runtime (WindowViewer) by designing a particular action and associating it to the event. For example, if your ActiveX control has an error event handler, you could create a ActiveX Event script that displays a window with an error message when an error occurs or any other OpenHMI QuickScript. ActiveX Event scripts are provided to support event actions. You can associate a named event script to each event.

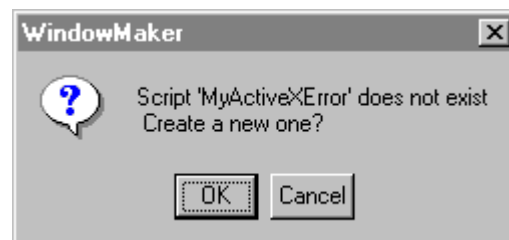
**Note** In order for an ActiveX Event script to function properly the ActiveX control for which the script was created, must be loaded into memory. If the window containing an ActiveX control is closed, its ActiveX Event scripts, or any other OpenHMI QuickScripts containing script functions associated with that ActiveX control, will not execute properly.

➤ **To use ActiveX Event parameters:**

1. Double-click the ActiveX control for which you want to create an ActiveX Event script. The selected ActiveX control's **Properties** dialog box will appear.
2. Click the **Events** tab to activate the **Events** property sheet:




3. In the **Event** column select the event to which you want to associate an ActiveX Event Script.
4. In the respective cell in the **Script** column, type a unique name for the ActiveX Event Script that you want to create and then double-click the name, or click **OK**. The following message box appears:

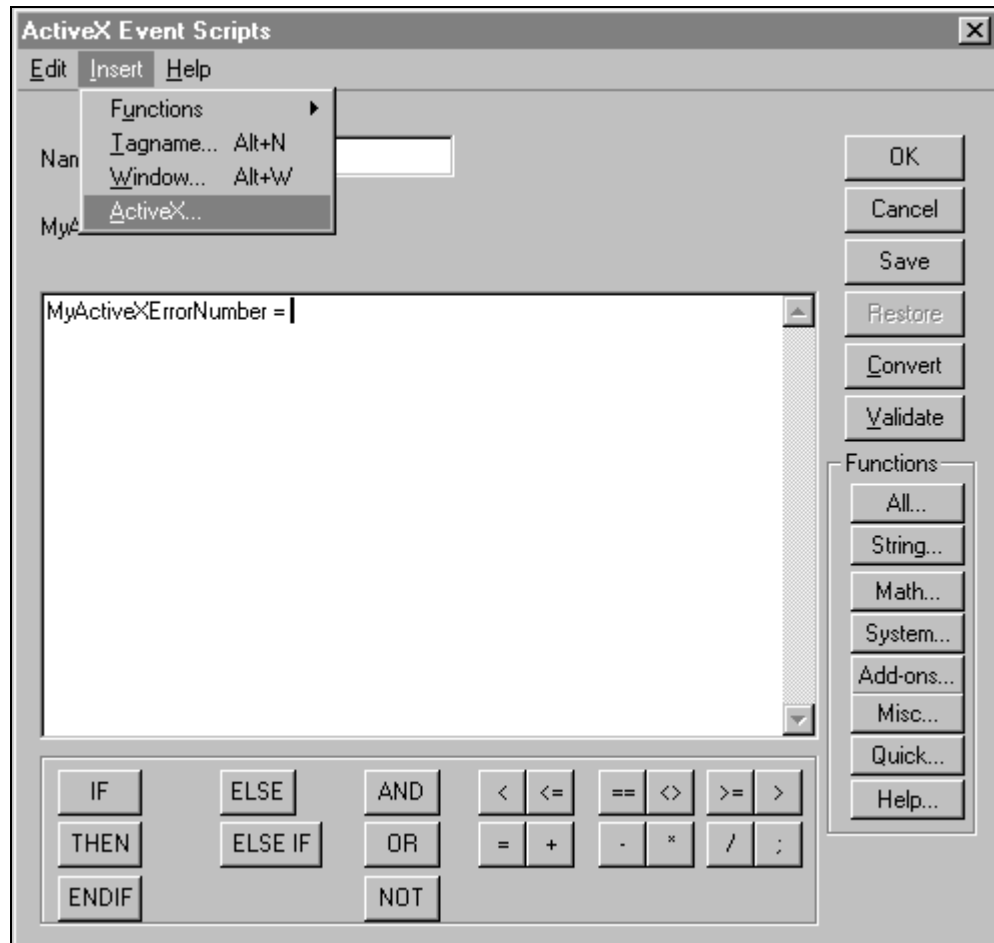


Click **OK**. The ActiveX Event script editor will appear displaying the name that you typed in the **Name** input box (see example below). If you double-click a

blank **Script** cell, when the ActiveX Event script editor appears, you must then type a name for the ActiveX Event script.

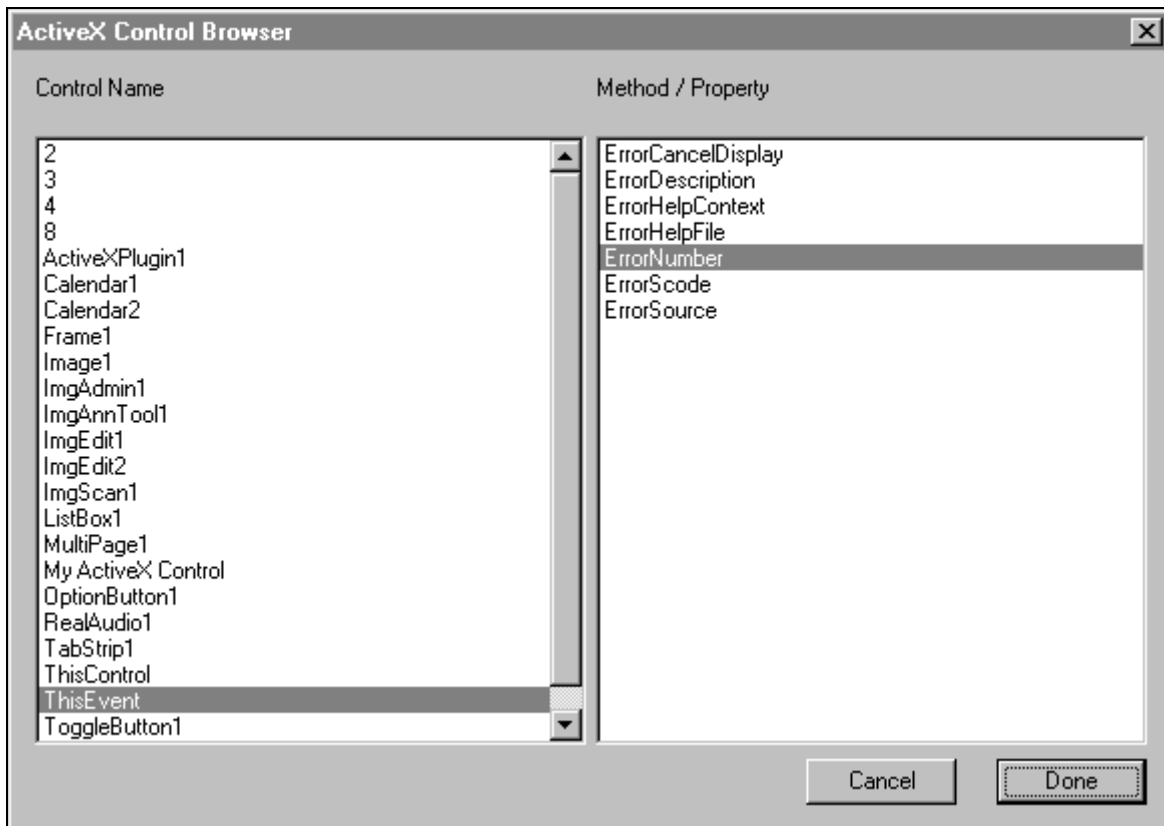
- ☞ If the ActiveX Event script that you want to use already exists, click the  button. The **Choose ActiveX Script** dialog box will appear listing all existing ActiveX Event scripts in your application.

☞ For more information, see "[Reusing ActiveX Event Scripts.](#)"



5. On the **Insert** menu, click **ActiveX**. The **ActiveX Control Browser** appears:





- In the **Control Name** list, select **This Event** to access the parameters for the selected event. In this case, the selected event is, **Error**.

---

**Note** **This Event** is accessible only through ActiveX Event scripts. It is not accessible through any other type of OpenHMI QuickScript. You must select **This Event** to access the event parameters for an ActiveX control.

Events may or may not pass parameters in runtime. Event parameters can be accessed by using the **ThisEvent** keyword. For example:

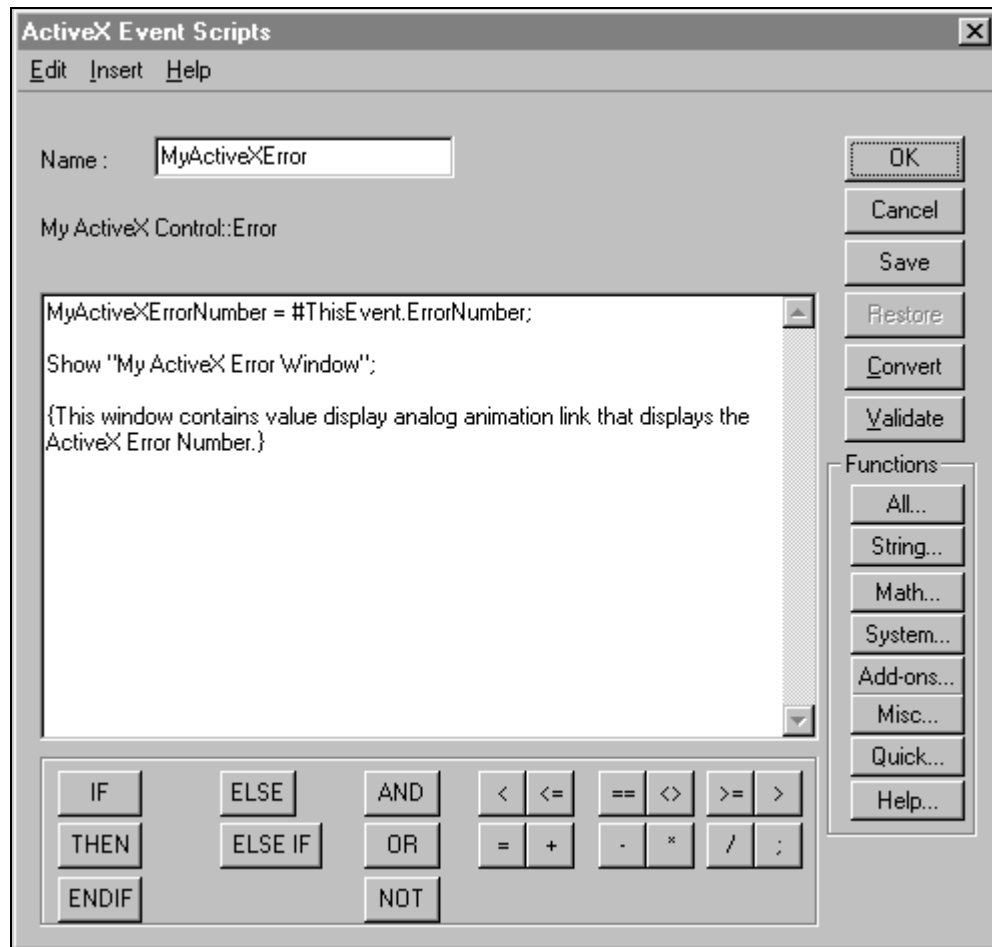
```
MyActiveXErrorNumber = #ThisEvent.ErrorNumber;
```

Where: # indicates that this is an ActiveX script function. **ThisEvent** relates to the event selected in the ActiveX control's **Event** property sheet, and **ErrorNumber** is the parameter passed by the selected event.

---

- In the **Method / Property** list, select the event that you want to use in your ActiveX Event script.
- Click **Done**.

The selected control name, in this case, **This Event**, and selected event parameter are both automatically inserted into your script at the cursor location. For example:

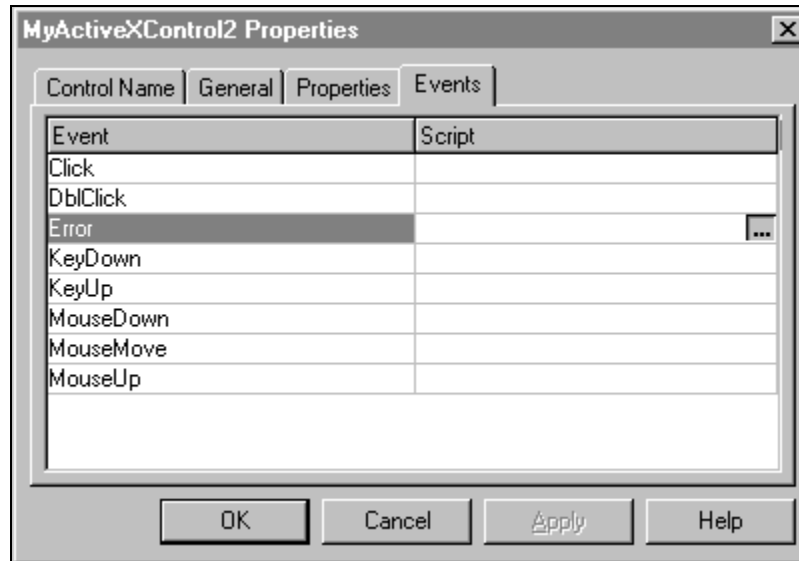


9. Click **OK** to save your ActiveX Event script and close the script editor. The ActiveX control's **Properties** dialog box reappears.
10. Click **OK** to close the **Properties** dialog box, or continue to create ActiveX Events scripts.

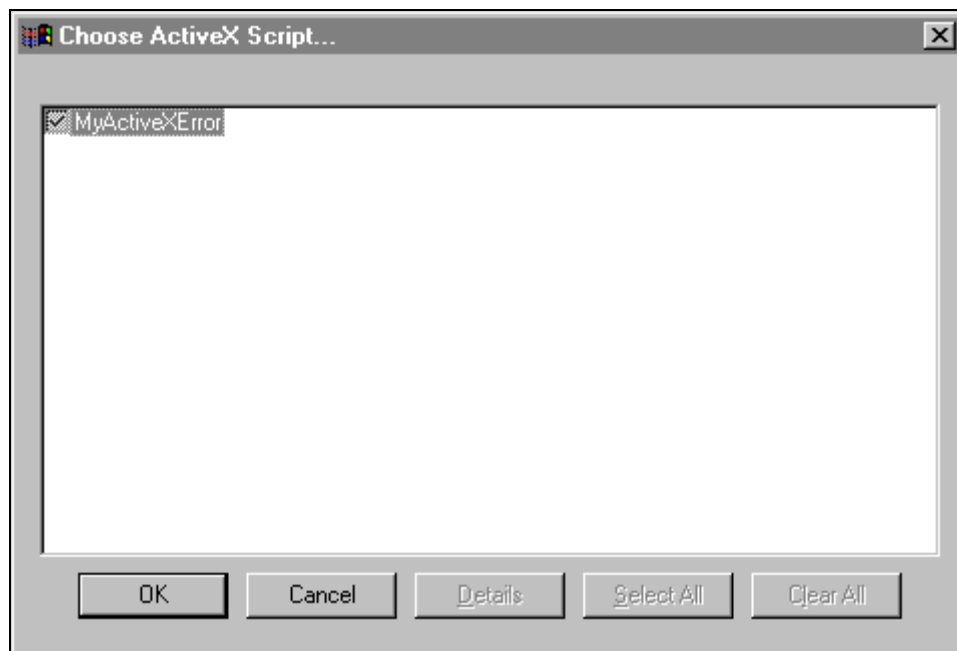
## Reusing ActiveX Event Scripts

ActiveX Event scripts can only be reused for the same event for the same kind of ActiveX control. For example, the mouse down event may be a stock event on hundreds of ActiveX controls. However, an ActiveX Event script written for mouse down on ActiveX ControlA cannot be reused for mouse down on ActiveX ControlB unless the two controls are the same type.

- **To reuse an ActiveX Event script:**
  1. Double-click the ActiveX control for which you want to reuse an existing ActiveX Event script. The selected ActiveX control's **Properties** dialog box will appear.
  2. Click the **Events** tab to activate the **Events** property sheet:




- In the **Script** column for the respective event, click the button. The **Choose ActiveX Script** dialog box appears:

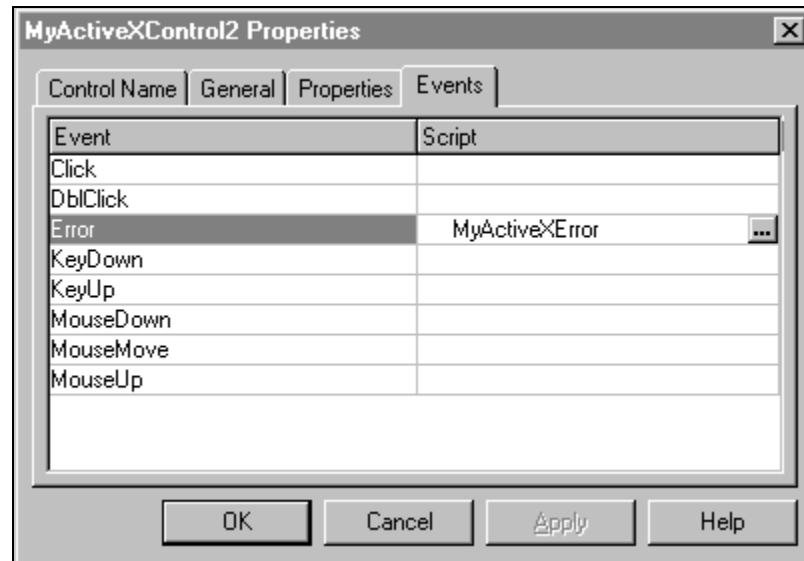


**Note** This dialog box will only display the ActiveX Event scripts that were written for the same type of ActiveX control and the same selected event.

For example, let's assume that you are creating an ActiveX Event script for a second ActiveX Calendar control's "Click" event. You have already created two other ActiveX Event scripts named Click1 and Click2 in your application. Click1 was created for a different ActiveX Calendar control's "Click" event, and Click2 was created for an ActiveX InSQLTrend control's "Click" event. When you click the button and the **Choose ActiveX Script** dialog box appears, it will only display the Click1 script since it was created for the same type of ActiveX control and the same event.

- Select the ActiveX Event script that you want to use, and then click **OK**.

The name of the selected script is automatically inserted into the **Script** cell where you previously clicked the  button. For example:



5. Click **OK** to close the **Properties** dialog box, or continue to create ActiveX Events scripts.


## Importing ActiveX Event Scripts

Importing ActiveX Event scripts from one OpenHMI application to your current application, can save you a considerable amount of development time. When you move ActiveX Event scripts from one OpenHMI application to another, you must use the **Import** command on the WindowMaker **File** menu.

---

**Note** When you import ActiveX Event scripts, from one application to another, all ActiveX Events scripts are imported. Additionally, in order for an imported ActiveX Event script to function properly in the new application, the same ActiveX control and the same event for which the script was originally created, must also be used in the new application, and it must be loaded into memory. If the window containing an ActiveX control is closed, its ActiveX Event scripts, or any other OpenHMI QuickScripts containing script functions associated with that ActiveX control, will not execute properly.

---

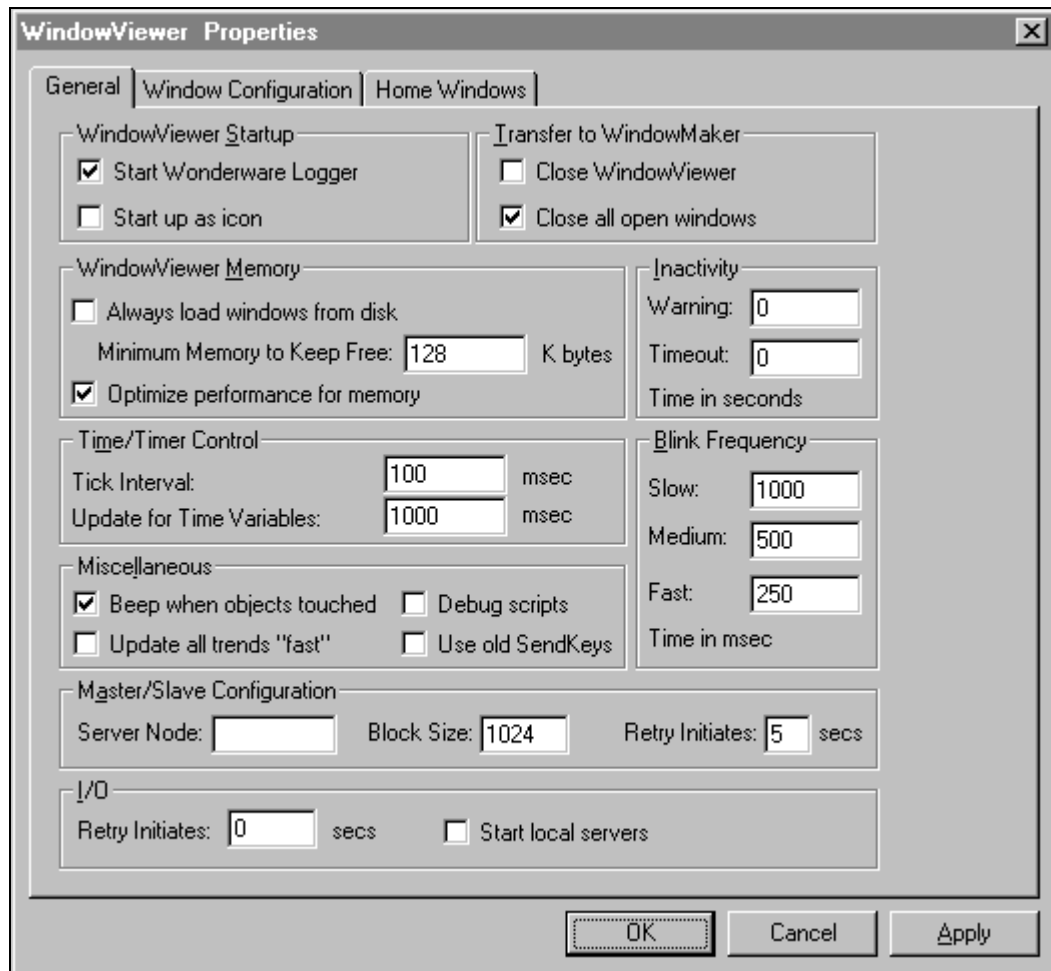
 For more information on importing scripts, see Chapter 5 Creating QuickScripts in OpenHMI.

# Customizing Your Runtime Environment

Like WindowMaker, there are many properties that you can set to customize your runtime environment (WindowViewer). For example, you can set the blinking speed for blinking objects, the system inactivity timeout and warning values, the windows that are automatically opened when WindowViewer is started from its icon or its menu command.

## Setting WindowViewer's General Properties

- **To set the properties for WindowViewer:**
  1. On the **Special** menu, point to **Configure**, and then click **WindowViewer**, or in the Application Explorer under **Configure**, double-click **WindowViewer**. The **WindowViewer Properties** dialog box appears with the **General** properties sheet active:
    - ☞ In the Application Explorer, you can also right-click **WindowViewer**, and then click **Open**.



- ☞ If you right-click any of the text entry boxes in any dialog box, a menu will appear displaying the commands that you can apply to the selected text.

2. Select **Start Wonderware Logger**, if you want the Logger program to automatically start when you start WindowViewer.
  - ☞ The Logger's behavior is a little different when you are on the Windows NT Operating System. For more information on the Logger, see "[Welcome to OpenHMI.](#)"
3. Select **Start up as icon** if you want WindowViewer to start up as an icon instead of a window.
  - ☞ Select this option only when you are using WindowViewer to gather data for other I/O-interconnected applications.
4. Select **Close WindowViewer** if you want WindowViewer to automatically close when you start WindowMaker.
  - ☞ If memory is not an issue, and you are using the fast switch to move between WindowViewer and WindowMaker, this option should not be selected.

The fast switch option is selected in the **WindowMaker Properties - General** dialog box.

When you select this option, the **Close on Transfer to WindowViewer** option located on the **WindowMaker Properties/General** property sheet is automatically selected too.

5. Select **Close all open windows** if you want all open windows to automatically be closed when you transfer from WindowViewer to WindowMaker.
  - ☞ Selecting this option will free up memory on your system.
6. Select **Always Load Windows from Disk** if a low memory situation exists.
  - ☞ Selecting this option causes your application windows to be loaded from disk and not saved in RAM memory when you close them.
7. In the **Minimum Memory to keep free** box, type the number of K bytes of memory that you want to keep free for other Windows applications.
8. Select **Optimize performance for memory** to significantly increase the drawing update speed. Selecting this option significantly increases the update rate for text fields.
  - ☞ If your system is low on memory do not enable this option.
9. In the **Warning** box, type the number of seconds that can elapse with no operator activity (mouse clicks or keystrokes) before the system discrete tagname **\$InactivityWarning** is set to 1 (True).
  - ☞ You can use **\$InactivityWarning** in a Condition QuickScript to show a window warning the operator that he/she is about to be logged off the system. If the operator clicks the mouse, presses a key, or performs an action using any other pointing device before the specified timeout elapses, they are not logged off. **\$InactivityWarning** and the timer are reset.
10. In the **Timeout** box, type the number of seconds that can elapse with no operator activity (mouse clicks, keystrokes, and so on) before the system discrete tagname **\$InactivityTimeout** is set to 1 (True). When **\$InactivityTimeout** is true, the system equates the logged on operator name to the reserved name "None" and sets the security tagname, **\$AccessLevel**, to 0.
  - ☞ You can use **\$InactivityTimeout** in a Condition QuickScript to show a window telling the operator that he/she has been logged off the application.

You can use the **Timeout** feature independently of the **Warning** feature. However, the **Timeout** value must be greater than the **Warning** value for proper use of both system tagnames.

For example, **Warning** becomes true after 30 seconds of inactivity and **Timeout** becomes true after an additional 15 seconds (for a total of 45 seconds) of inactivity.

11. In the **Tick Interval** box, type the speed interval that OpenHMI will use to check its internal timers.

☞ This setting controls how fast an Application **While Running**, Window **While Showing**, Condition **While On True/On False**, Key and Touch Pushbutton Action **While Down** QuickScripts will be executed.

---

**Note** Scripts cannot execute faster than every 10 milliseconds on the Windows NT operating system or every 50 milliseconds on Windows 95.

---

☞ For more information, see Chapter 5 - Creating QuickScripts in OpenHMI.

12. In the **Update Time Variables every** box, type the frequency (in milliseconds) that you want WindowViewer to update the time-based system tagnames such as \$Msec, \$Second, \$Minute, and so on.

☞ We recommend that you use the default setting of 1000 milliseconds. However, you can type a zero to prevent updating of all time variables.

13. Select **Beep when objects touched** if you want all touch-sensitive objects to beep when selected in WindowViewer.
14. Select **Update all trends "Fast"** if you want your trend objects to be updated faster.

☞ Select this option only when you are absolutely certain that no objects are overlapping your runtime trend objects. If you select this option and any object is overlapping the trend object, it will not be drawn properly.

15. Select **Debug Scripts** if you want a message to be written to the Logger program each time a QuickScript is executed.

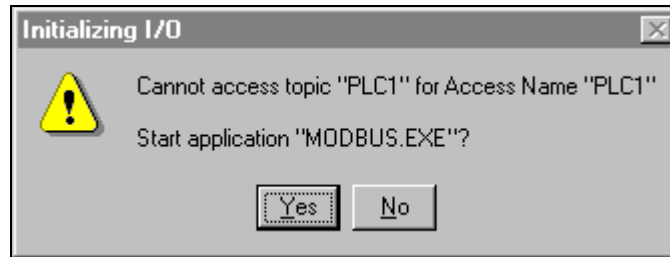
☞ If you select the **Debug** menu option in the **WindowViewer Properties/Window Configuration** property sheet, you will be able to turn this command on and off in runtime from WindowViewer's **Special** menu.

---

**Note** **Use old SendKeys** is a legacy option and is not used for OpenHMI.

---

16. Select **Use old SendKeys** only if you are using an international application that was developed using OpenHMI Version 3.26 or earlier. (This is a legacy option and it is not used for OpenHMI.)
17. In the **Slow**, **Medium**, **Fast** boxes type the speeds (in milliseconds) that you want to use for your blink animation links.
22. Select the **Start Local Servers** option if you want a dialog box to display whenever you start WindowViewer and the server you are trying to communicate with is not running. For example:



Click **Yes** to start the server or, click **No** to ignore the message and close the dialog box.

23. Click **OK** to save your property settings and close the dialog box.

---

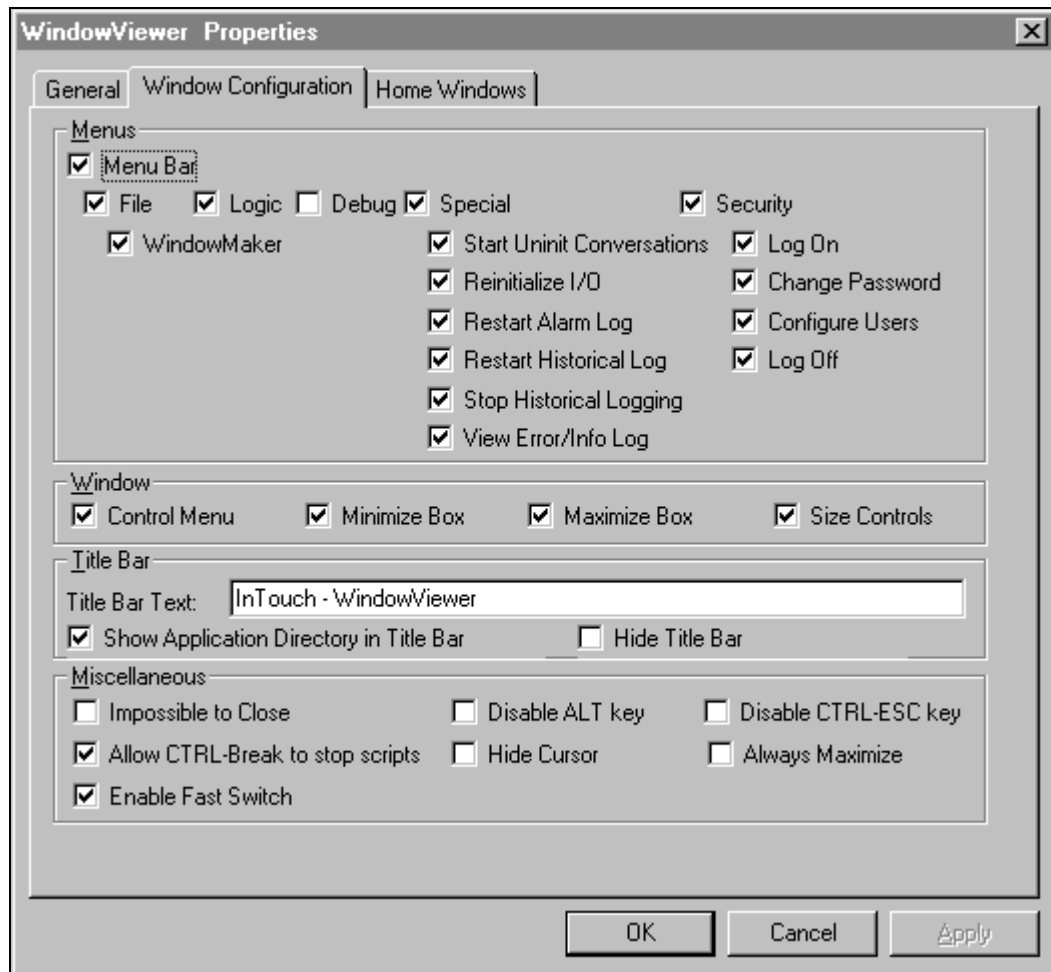
**Note** After you modify any of these parameters, you must restart WindowViewer to apply your changes.

---

## Setting WindowViewer's Window Configuration Properties

- **To configure the WindowViewer program window:**
  1. On the **Special** menu, point to **Configure**, and then click **WindowViewer**, or in the Application Explorer under **Configure**, double-click **WindowViewer**. The **WindowViewer Properties** dialog box appears:
    - ☞ In the Application Explorer, you can also right-click **WindowViewer**, and then click **Open**.
  2. Click the **Window Configuration** tab:





☞ If you right-click any of the text entry boxes in any dialog box, a menu will appear displaying the commands that you can apply to the selected text.

3. In the menus group, select the menus and commands that you want displayed in runtime.

☞ By default, the menu bar is displayed when WindowViewer is running. Clear the **Menu Bar** option to prevent the menu bar from showing.

Clear the **WindowMaker** command if you want to prevent the operator from being able to switch to the WindowMaker program. (Selecting this option does not affect the fast switch to WindowMaker.)

Clear the **Logic** menu if you want to prevent the operator from starting and stopping all QuickScripts from executing during runtime.

You can use the system tagname, **\$LogicRunning** to allow the operator to start and stop all QuickScripts.

---

**Note** If you select the **Allow CTRL-Break to stop scripts** option, the operator will be able to stop all QuickScripts from executing regardless of whether the **Logic** menu is displayed or not.

---

Select the **Debug** menu only when you need to "debug" your application.

Select the **Window** controls that you want available in runtime.

---

**Note** You must clear the **Control Menu** option (also called the System menu) in order to hide the close (X button) in the upper right hand corner of the application.)

---

4. In the **Title Bar Text** box, type the title that you want to appear in your application's title bar in runtime. For example:

**ABC Company, Paint APP1**

---

**Note** You cannot change the title bar if you are using a "Promotional License."

---

5. Select **Show Application Directory** if you want to include the path to the application's directory in the title bar. For example:  
**ABC Company, Paint APP1 - C:\DEMOAPP1**
6. Select **Hide Title Bar** if you want to hide the application's title bar in runtime.
7. Select **Impossible to Close** if you want to prevent the operator from closing WindowViewer.

---

**Note** You must clear the **Control Menu** option (also called the System menu) in order to hide the close (X) box in the upper right hand corner of the application.)

---

8. Select **Allow CTRL-Break to stop scripts** if you want to allow the operator to press the CTRL + BREAK key sequence to stop the execution of all QuickScripts whenever necessary during runtime.
9. Select **Disable ALT key** if you want to disable the ALT key and prevent the operator from executing menu commands by using the ALT + accelerator key. For example, ALT + F4 to exit the application.

---

**Note** You must clear the **Control Menu** option (also called the System menu) in order to hide the close (X) box in the upper right hand corner of the application.)

---

10. Select **Hide Cursor** if you want to prevent the cursor from being displayed during runtime because a touch-screen will be used.
11. Select **Disable CTRL-ESC key** if you want to prevent the operator from accessing the Windows **Start** menu to close and/or switch applications.
12. Select **Always Maximize** if you want to keep the WindowViewer program maximized at all times.
13. Click **OK** to save your settings and close the dialog box.

---

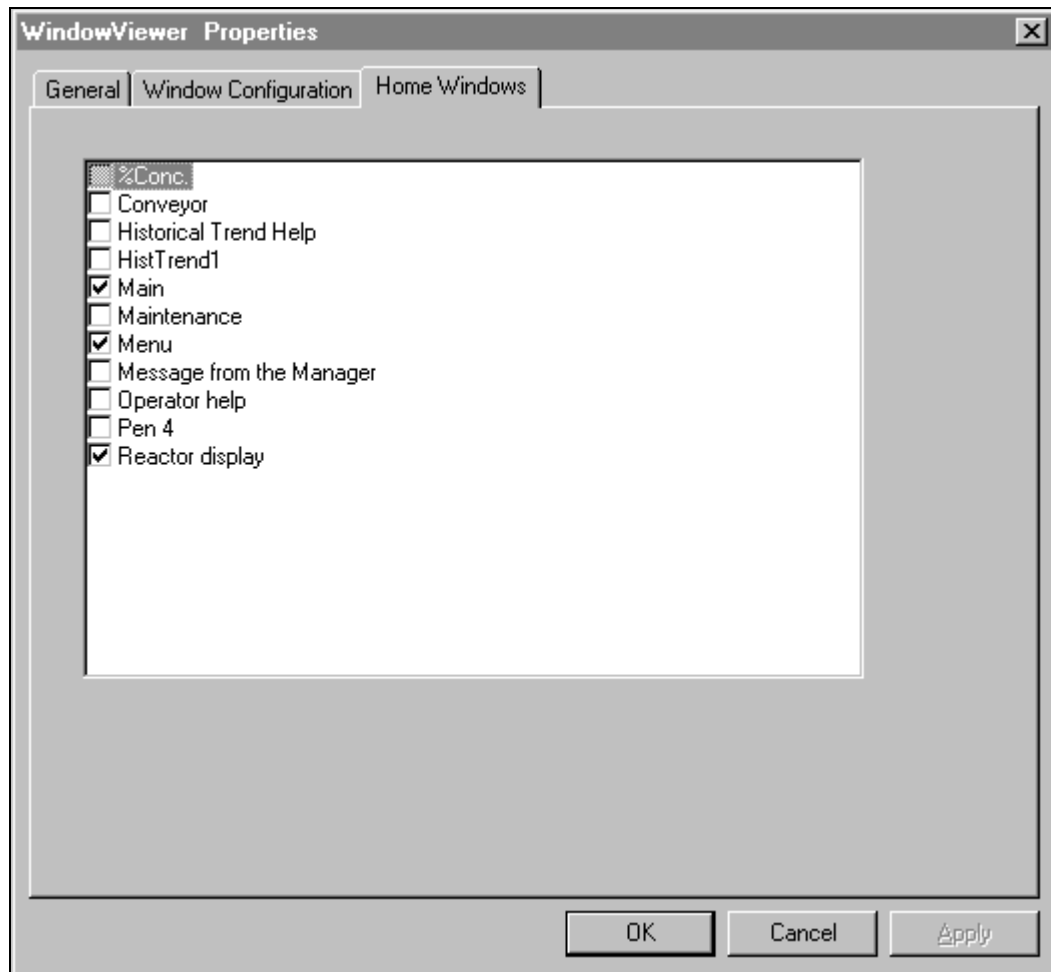
**Note** After you modify any of these parameters, you must restart WindowViewer to apply your changes.

---

## Selecting WindowViewer's Home Windows

➤ **To select the WindowViewer default start up windows:**

1. On the **Special** menu, point to **Configure**, and then click **WindowViewer**, or in the Application Explorer under **Configure**, double-click **WindowViewer**. The **WindowViewer Properties** dialog box appears:
  - ☞ In the Application Explorer, you can also right-click **WindowViewer**, and then click **Open**.
2. Click the **Home Windows** tab:



3. Select the window(s) that you want to automatically open when WindowViewer is started directly.

---

**Note** The home windows selections have no effect when you use the fast switch to start WindowViewer. Home windows are automatically opened when you start WindowViewer directly from either its icon or its menu command.

---

4. Click **OK**.

## Using OpenHMI Security

Applying security to your application is optional. However, by applying security to your application, you can control specific functions that an operator is allowed to perform by linking those functions to internal tagnames. In addition, when you establish security on your application, audit trails can be created that tie the operator to all alarms/events that occur during the time he/she is logged on to the system.

Security is based on the concept of the operator "logging on" to the application, typing his/her name and his/her password. Therefore, you must configure a user name, password and access level for each operator.

---

**Note** There is no association between Microsoft operating system security and OpenHMI security.

---

When you create a new application, by default, the user name is set to "Administrator" with an access level of 9999 (which allows access to all security commands). Once you add a new user name to the security list and restart WindowMaker or WindowViewer, the default user name is automatically reset to "None" with an access level of "0" (which prevents access to the Configure Users command in both WindowMaker and WindowViewer). Therefore you must configure a user name for the System Administrator with an access level equal to or greater than 9000 in order to be able to access the security user list later.


Once an operator logs on to the application, access to any protected function will be granted upon verification of the operator's password and access level against the value specified for the internal security tagname linked to the function.


For example, you can control access to a window, or the visibility of an object and so on, by specifying that the logged on operator's "Access Level" must be greater than 2000.

---

**Note** The operator can log on to the application by executing the **Log on** menu command under **Security** in the WindowViewer **Special** menu (if the **Special** menu is displayed) or, you can create a custom log on window with touch-sensitive input objects that are linked to internal security tagnames.

---

 For more information on the internal security tagnames, see the *OpenHMI Reference Guide*.

 The commands used to establish security on an application located under **Security** on the **Special** menu in both WindowMaker and WindowViewer. The security commands are used to log on and off the application, change passwords and to configure the list of valid user names, passwords and access levels.

## Using the Security Internal Tagnames

Once you implement security on your application, there are two internal security tagnames that you can use on buttons, in animation link expressions or QuickScripts, and so on, to control whether or not the logged on operator is allowed to perform specific functions:

Tagname	Type	Valid Values	Access
\$AccessLevel	Integer	0-9999	Read Only
\$Operator	Message	16-characters max	Read Only

For example, to make an object become visible based on the logged on user's access level, the following statement could be used in a visibility animation link's

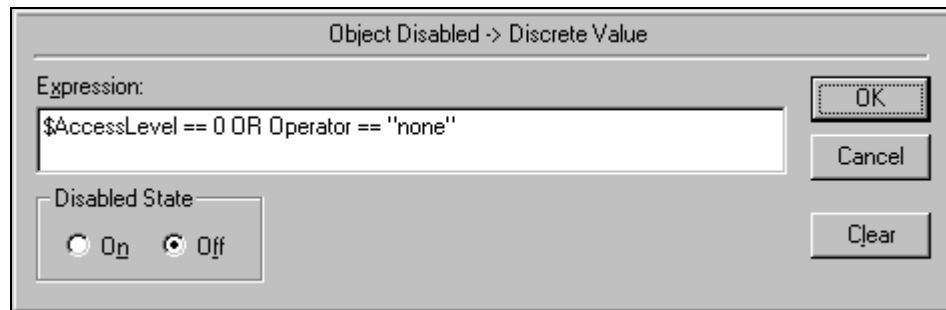
expression:

```
$AccessLevel >= 2000;
```

Or, a QuickScript can be bounded by an **IF** statement:

```
IF $Operator == "DayShift" THEN
  Show "Control Panel Window";
  {and other lines that only execute for the DayShift
   Operator}
ENDIF;
```

You can also control an object's touch functionality based upon the value of an internal security tagname by using the **Disable** animation link. For example:

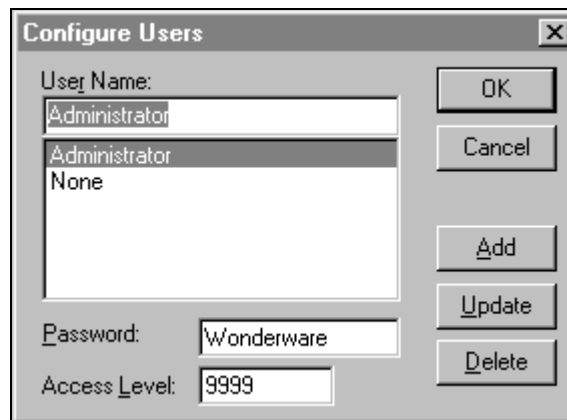


By using the above expression if no one is logged on, the object or button is secured from tampering.

## Configuring the Operator's Security Level

### ➤ To configure security for the operators of your application:

1. On the **Special** menu, point to **Security**, and then click **Configure Users**. The **Configure Users** dialog box appears:



- ☞ If you right-click any of the text entry boxes in any dialog box, a menu will appear displaying the commands that you can apply to the selected text.
2. In the **User Name** field, type the name that you want to assign to the operator.
  3. In the **Password** field, type a password (up to 15 characters).
  4. In the **Access Level** field, type a value (lowest = 0 to highest = 9999).
  5. Click **Add** to add the user name to the security list.
- ☞ To modify an existing user name, select the desired name in the **User Name** list. Type your changes, and then click **Update** to accept the changes. To delete a user name, select it in the list, and then click **Delete**.

---

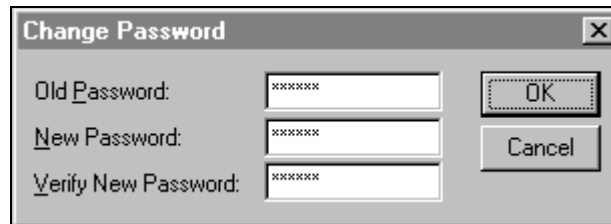
**Note** The **None** and **Administrator** names are reserved and only the password (**Wonderware**) or **Administrator** may be changed. Once you have configured user names for your application, you should change the **Administrator** name's password since it will more than likely become commonly known to most users

of the system. The **Administrator** default access level (9999) is the highest and allows access to everything including, the **Configure Users** menu command.

## Changing a Security Log On Password

➤ **To change the password for an operator:**

1. On the **Special** menu, point to **Security**, and then click **Change Password**. The **Change Password** dialog box appears:



☞ If you right-click any of the text entry boxes in any dialog box, a menu will appear displaying the commands that you can apply to the selected text.

2. In the **Old Password** field, type the old password.
3. In the **New Password** field, type the new password (up to 16 characters).
4. In the **Verify Password** field, type the new password again.
5. Click **OK**.

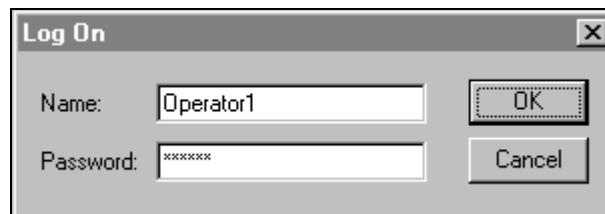
☞ To prevent anyone who may be watching the operator from seeing the password, the information entered is displayed on the screen as asterisks.

**Note** If you do not plan on displaying the **Special** menu in WindowViewer, you can create a discrete button and link it to the **\$ChangePassword** internal tagname to set the **\$ChangePassword** tagname equal to 1 to cause the **Change Password** dialog box to be displayed. Once displayed, the operator can change his/her password.

## Logging on to an Application

➤ **To "log on" to an application:**

1. On the **Special** menu, point to **Security**, and then click **Log On**. The **Log On** dialog box appears:



☞ If you right-click any of the text entry boxes in any dialog box, a menu will appear displaying the commands that you can apply to the selected text.

2. In the **Name** box, type your user name.
3. In the **Password** box, type your password.
4. Click **OK**.

- ☞ If the information is entered incorrectly or is invalid, a message box indicating that log on failed will appear.  
If log on is successful, the **\$AccessLevel** internal tagname will be set to its predefined value (configured in the security user list).

## Creating a Custom Security Log on Window

If the **Special** menu will not be displayed in WindowViewer, you can create a custom log on window that the operator uses to log on to the application.

➤ **To create a custom log on window:**

Link the **\$OperatorEntered** and **\$PasswordEntered** system tagnames to user input objects or, use them in a QuickScript to set the "User Name" and "Password." (These are internal message type tagnames that are intended for write operation only.)

For example:

```
Set the User Name string into ->    $OperatorEntered
```

```
Set the User Password string into -> $PasswordEntered
```

If the entries are valid, the **\$AccessLevel** and **\$Operator** internal tagnames are set to their predefined values (configured in the security user list). i

Also, when you are not displaying the **Special** menu in WindowViewer, you can link a **User Input - Discrete** button to the **\$ChangePassword** tagname to show the **Change Password** dialog box and allow the operator to change his/her password. When the operator clicks the button, the value of the **\$ChangePassword** tagname is set to 1 and the **Change Password** dialog box appears. When the operator closes the dialog box, the system resets the value to 0. (This is a system discrete tagname intended for write operation only.)

You can also link a **User Input - Discrete** button to the **\$ConfigureUsers** tagname to allow an authorized operator to access the **Configure Users** dialog box to edit the security user name list. When the operator clicks the button, the value of the **\$ConfigureUsers** tagname is set to 1 and the **Configure Users** dialog box appears. When the operator closes the dialog box, the system resets the value to 0. (This is a system discrete tagname intended for write operation only.)

## Logging Off an Application

➤ **To log off the application:**

On the **Special** menu, point to **Security**, and then click **Log Off**.

- ☞ When this command is executed, the "User Name" is reset to "None" with an Access Level of "0".

You can configure the application to automatically log off the operator after a specified amount of time has elapsed with no activity by the operator.

☞ For more information, see "[Automatically Logging Off the System.](#)"

## Automatically Logging Off the System

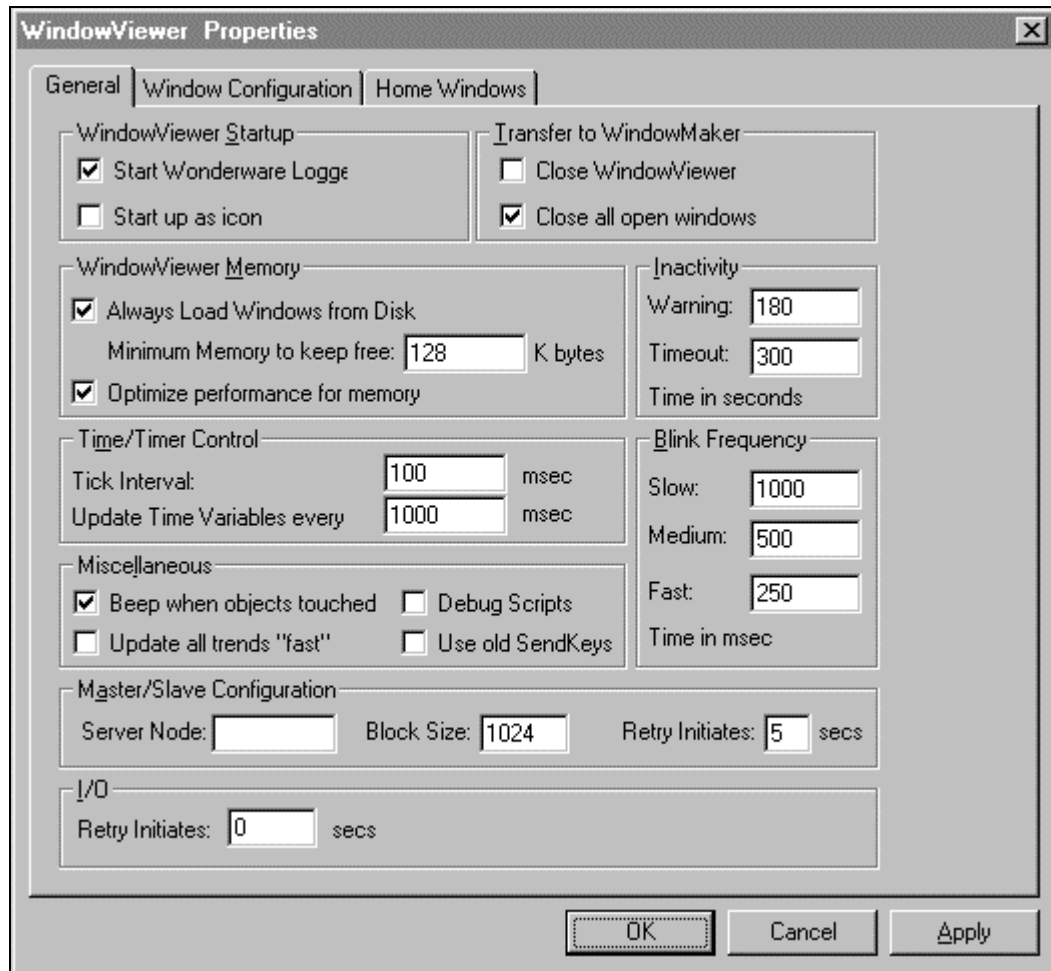
You can configure your application to automatically log off the operator when there has been no activity for a specified period of time by using the warning and timeout settings.



➤ **To configure inactivity:**

1. On the **Special** menu, point to **Configure**, and then click **WindowViewer**, or in the Application Explorer under **Configure**, double-click **WindowViewer**. The **WindowViewer Properties** dialog box appears with the **General** properties sheet active:

☞ In the Application Explorer, you can also right-click **WindowViewer**, and then click **Open**.



- ☞ If you right-click any of the text entry boxes in any dialog box, a menu will appear displaying the commands that you can apply to the selected text.
2. In the **Warning** box, type the number of seconds that can elapse with no operator activity (mouse clicks or keystrokes) before the system discrete tagname **\$InactivityWarning** is set to 1 (True).
 

☞ You can use **\$InactivityWarning** in a Condition QuickScript to show a window warning the operator that he/she is about to be logged off the system. If the operator clicks the mouse, presses a key, or performs an action using any other pointing device before the specified timeout elapses, they are not logged off. **\$InactivityWarning** and the timer are reset.
  3. In the **Timeout** box, type the number of seconds that can elapse with no operator activity (mouse clicks, keystrokes, and so on) before the system discrete tagname **\$InactivityTimeout** is set to 1 (True). When

**\$InactivityTimeout** is true, the system equates the logged on operator name to the reserved name "None" and sets the security tagname, **\$AccessLevel**, to 0.

☞ You can use **\$InactivityTimeout** in a Condition QuickScript to show a window telling the operator that he/she has been logged off the application.

You can use the **Timeout** feature independently of the **Warning** feature. However, the **Timeout** value must be greater than the **Warning** value for proper use of both system tagnames.

☞ **Example: Warning** becomes true after 30 seconds of inactivity and **Timeout** becomes true after an additional 15 seconds (for a total of 45 seconds) of inactivity.

4. Click **OK**.

☞ You can use **\$InactivityWarning** in a Condition QuickScript to show a window warning the operator that he/she is about to be logged off the system. If the operator clicks the mouse, presses a key, or uses his/her touch screen before the specified timeout elapses, they are not logged off. (**\$InactivityWarning** and the timer are reset.)

5. In the **Timeout** box, type the number of seconds that can elapse with no operator activity (mouse clicks, keystrokes, and so on) before the system discrete tagname **\$InactivityTimeout** is set to 1 (True). When **\$InactivityTimeout** is true, the system equates the logged on operator name to the reserved name "None" and sets the security tagname, **\$AccessLevel**, to 0.

☞ You can use **\$InactivityTimeout** in a Condition QuickScript to show a window telling the operator that he/she has been logged off the application.

You can use the **Timeout** feature independently of the **Warning** feature. However, the **Timeout** value must be greater than the **Warning** value for proper use of both system tagnames.

**Example: Warning** becomes true after 30 seconds of inactivity and **Timeout** becomes true after an additional 15 seconds (for a total of 45 seconds) of inactivity.

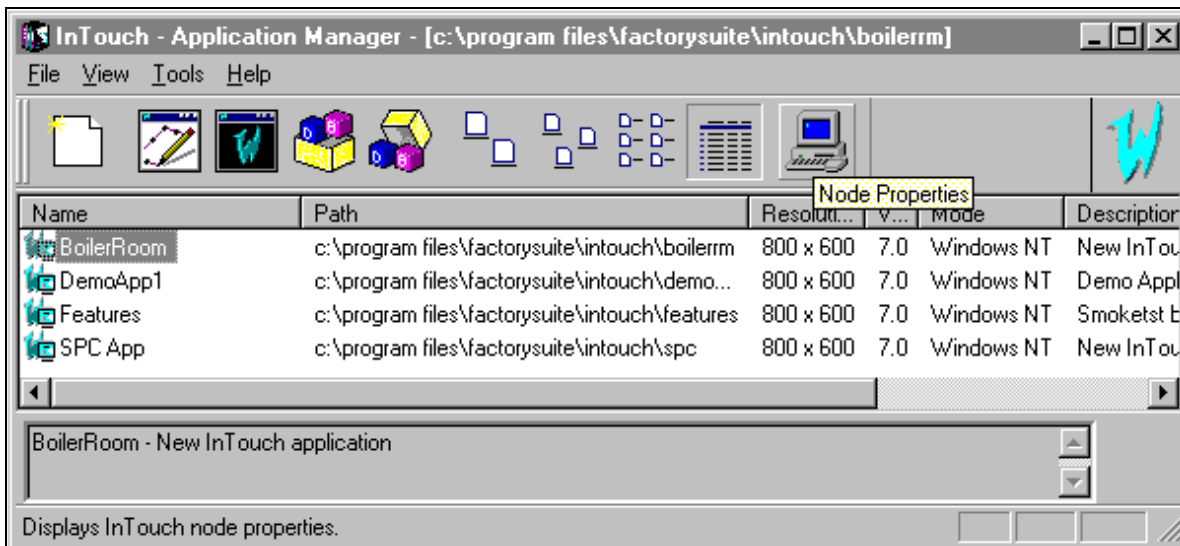
## Dynamic Resolution Conversion (DRC)

Dynamic Resolution Conversion(DRC) provides independence from screen resolution restrictions. DRC enables each OpenHMI workstation to scale the application to a number of user-defined options, including a custom resolution. This scaling takes place while WindowViewer compiles the application, and does not require WindowMaker. Since each workstation can use a different DRC setting, each must have its own settings configured.



➤ **To configure an application for DRC:**

1. Start the **Application Manager** dialog box appears:



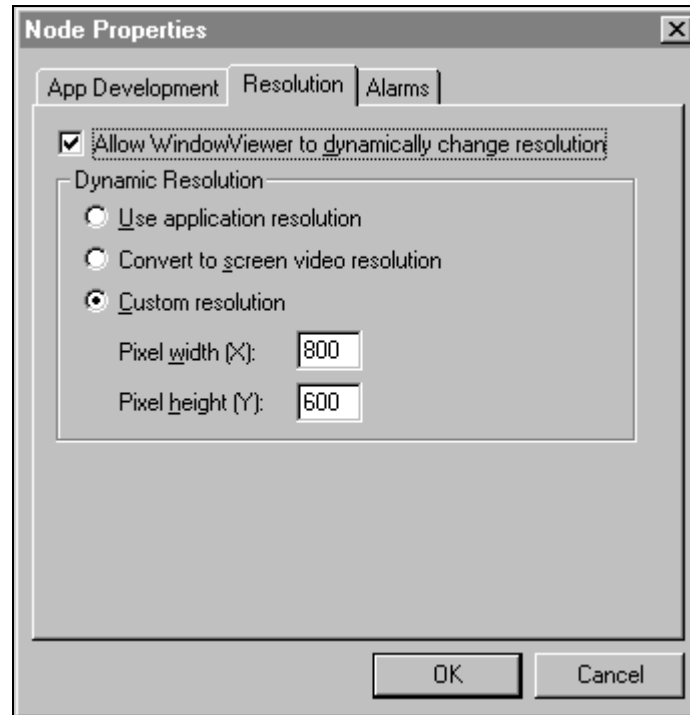
3. Click the **Node Properties** tool or on the **File** menu, click **Properties**. The **Node Configuration** dialog box will appear.
  - ☞ To quickly access the dialog box, right-click a blank area of the application display window, and then click **Properties**.

---

**Note** When an application is selected in the Application Manager window, selecting the **Properties** command on the **File** menu will display the **Properties** dialog box for that application.

---

4. Click the **Resolution** tab:



5. Select **Allow WindowViewer to dynamically change resolution** if you want WindowViewer to locally scale the master application, based on the resolution option you select. (The three resolution options are described below.)
  - ⌘ If you do not select this option, WindowViewer will only run the application if the node's screen resolution is identical to the screen resolution of the application development node. If the resolutions are different, WindowViewer prompts the operator to run WindowMaker to convert the application to the node's resolution. Use caution when doing this if you have set up a UNC path to the master application directory as this will only modify the original application.
6. Select **Use Application resolution** if you want WindowViewer to run the application at the resolution it was developed for and ignore the node's resolution. For example, if the application was developed at 640x480 and the node's resolution is 1024x768, WindowViewer will not dynamically scale the application. Instead, the application will be displayed at 640x480.
7. Select **Convert to screen video resolution** if you want WindowViewer to run the application at the node's resolution and ignore the resolution the application was developed at. For example, if the node is running at 640x480 and the application was developed at 1280x1024, WindowViewer will dynamically scale the application (smaller) to fit the node's 640x480 display. (This will more than likely be the most commonly used setting.)
8. Select **Custom Resolution** if you want WindowViewer to run the application at the resolution you specified in the **Pixel width (X)** and **Pixel height (Y)** (must be integer values) boxes. The application's resolution and the node's resolution are both ignored. For example, if **Pixel width (X)** and **Pixel height (Y)** are set to 512 and 384, respectively, the application will dynamically be scaled to fit in a 512x384-pixel area on the node's display.
9. Click **OK**.

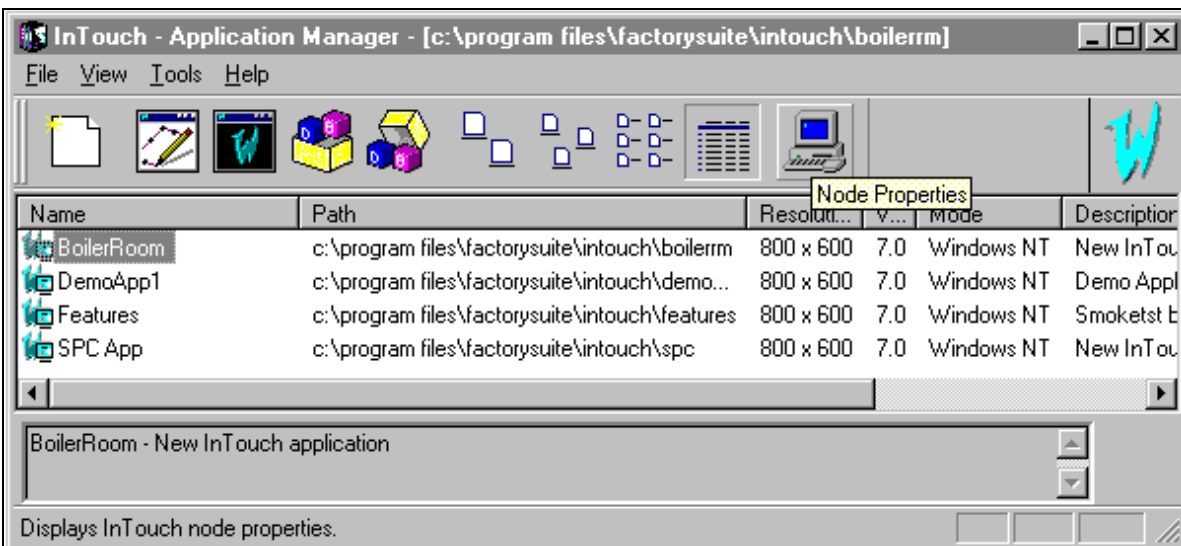
## Running WindowViewer as an NT Service

Running WindowViewer as an NT Service allows you to take advantage of all the features that an NT Service provides. For example, continuous operation after the operator logs off and automatic startup at system boot time without operator intervention. This allows unmanned station startup of WindowViewer without compromising NT operating system security.



➤ **To configure WindowViewer to start as an NT service:**

1. Start the OpenHMI program (OPENHMI.EXE). The **OpenHMI Application Manager** dialog box appears:



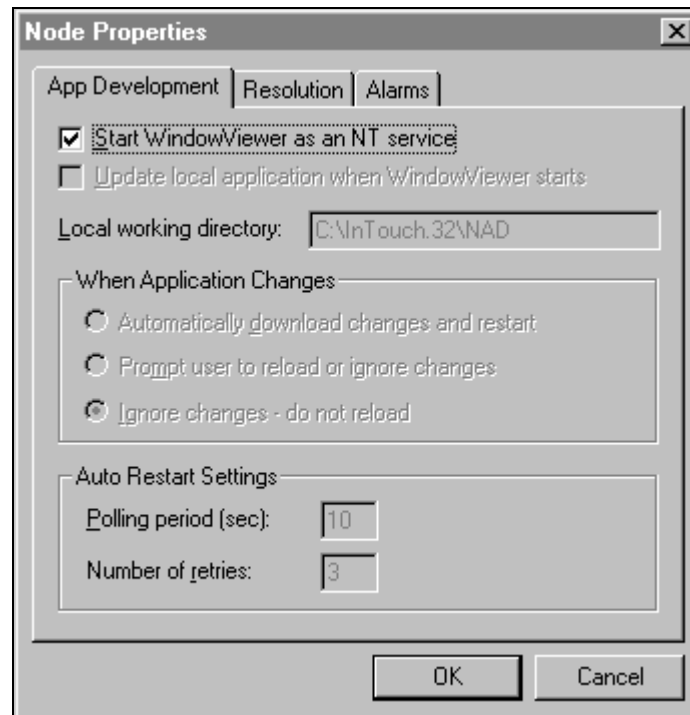
2. Click the Node Properties tool or, on the **File** menu, click **Properties**. The **Node Configuration** dialog box appears with the **App Development** property page active:

☞ To quickly access the dialog box, right-click a blank area of the Application Manager's window, and then click **Properties**.

---

**Note** When an application is selected in the Application Manager window, selecting the **Properties** command on the **File** menu will display the **Properties** dialog box for that application.

---

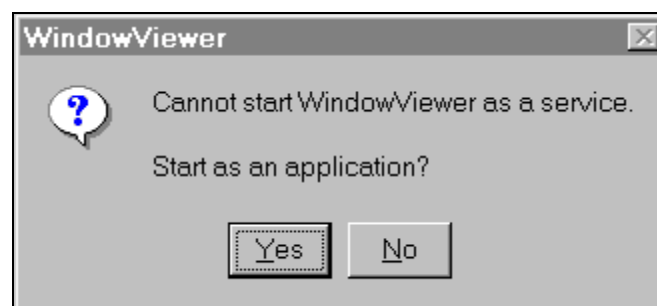


3. Select the **Start WindowViewer as an NT service** option to configure WindowViewer to automatically run as an NT service.
4. Click **OK**.

---

#### Notes

1) If WindowViewer is configured as an NT service and subsequently started directly (from its icon, the Windows startup menu and so on), there will be approximately a 15 second delay before WindowViewer will display a window. This delay is due to WindowViewer attempting to connect to the NT Service Control Manager. Upon failing to connect to the Service Control Manager, WindowViewer will display the following message box:



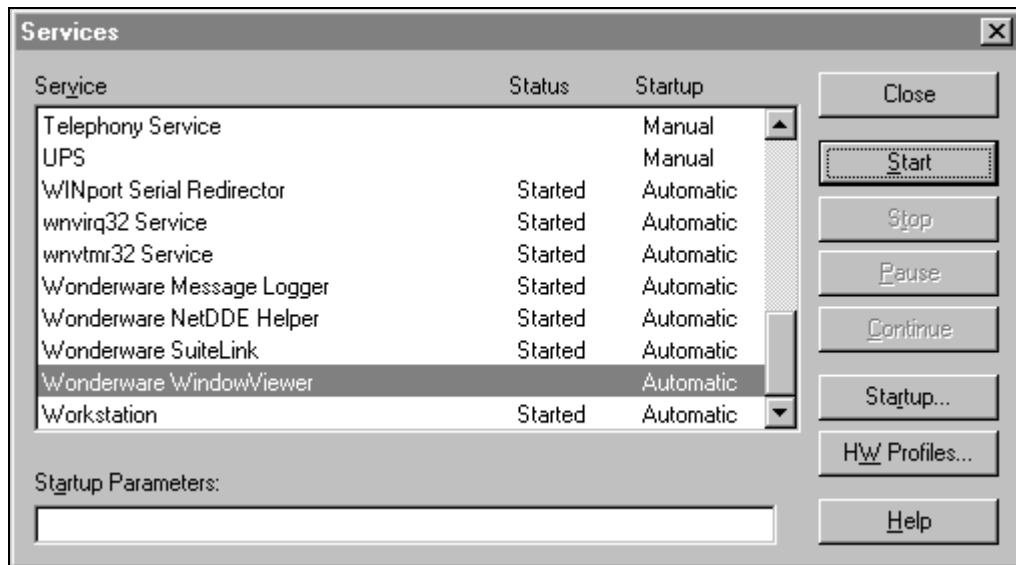
If you click **Yes**, WindowViewer is started as an application not an NT service. If click **No** the command to start WindowViewer is canceled.

2) If you stop WindowViewer from running as an NT service by clearing the **Start WindowViewer as an NT service** option, WindowViewer service is automatically uninstalled. However, it can be run as an application.

---

#### ➤ To start WindowViewer service from Services dialog box:

1. In the Windows Control Panel, double-click **Services**. The **Service** dialog box appears:



2. Select **Wonderware WindowViewer**, and then click **Start**.
3. Click **Close**.

☞ After you perform these steps, WindowViewer can be started as both an NT service and as an application.

## Configuring System Privileges

During OpenHMI installation, you will be prompted to provide your user name and password for your administrative account. This information is used to set up your NT user impersonation account. The Wonderware services such as, Wonderware NetDDE Helper and Wonderware WindowViewer, will use this information for automatically logon, and for automatically starting the appropriate services during unattended start up.

- ☞ The User Name and Password that you provide must be a valid Administrator level logon configured through the NT User Manager.

1. In the **Domain/Machine** box, type the name of your system domain or the node name.
2. In the **User Name** box, type your user identification.
3. In the **Password** box, type your system password.
4. In the **Confirm Password** box, retype you system password to verify it.

- ☞ After installation, if you need to alter this information, run the Wonderware Service User application (WWUSER.EXE) located in your installed directory. For example, \Program Files\FactorySuite\Common. When you run this utility, the **Wonderware Service User** dialog box appears:



---

Enter the information as described above.

## Time Zones

OpenHMI provides services that ease the use of applications across multiple time-zones. These services are used by both the alarm and history systems to permit values to be viewed at the local time they occurred. For example, if an engineer in California is viewing an alarm that occurred in a manufacturing plant in Kansas at 10AM, that engineer would see the local California time that the alarm occurred; 8AM. The same is true if the engineer is viewing historical data from that plant.

The key to these services is the use of UCT (Universal Coordinated Time), also known as Greenwich Mean Time or GMT, as the time reference. Each computer is configured with both the local time and a UCT offset for the time zone where it resides. In the above example, the time zone for the computer in California is set at GMT eight hours, while the time zone for the computer in Kansas is set at GMT six hours.

OpenHMI uses these GMT offsets as the basis for retrieving all alarm and historical data. In the above example, when the OpenHMI application in California received the alarm from the application in Kansas, it also looked at the GMT offset of both computers to determine the local California time that the alarm occurred. Thus, a 10AM alarm in a UCT six time zone equals an 8AM alarm in a GMT eight time zone. To use this feature, the GMT offset must be configured for each computer.

➤ **To set your time zone when using the Windows 95 operating system:**

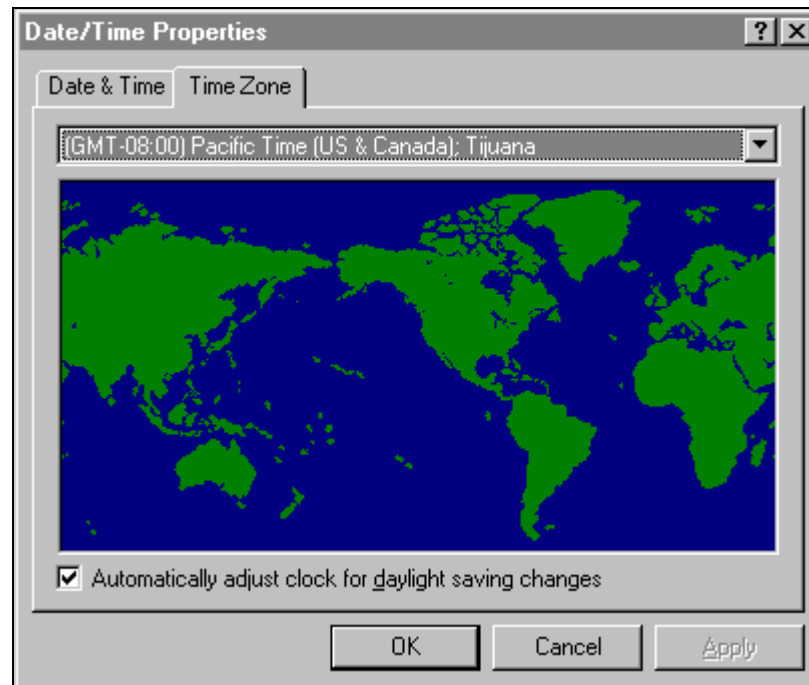
The setting, **set TZ=GMT|+ | -|X**, must be included in the AUTOEXEC.BAT of the computer.

Where: *X* is the offset from GMT for the time zone the computer resides in.)

For example, to set the **TZ** environment variable to correspond to the current time zone in California, you could use either **set TZ=GMT8** or **set TZ=GMT+8**.

➤ **To set your time zone on the Windows NT operating system:**

1. Open the Windows Control Panel.
2. Double-click the **Date/Time** icon or double-click the clock in the Windows **Taskbar**. The **Date/Time Properties** dialog box appears:



3. Click the **Time Zone** tab, and then click the arrow to open the list of time zones.
4. Select your time zone in the list.
5. Click **OK**.

---

**Note** The Windows 95 and Windows NT operating systems may be set to automatically adjust the clock for daylight savings time. It is recommended that this feature be disabled. To disable this feature, use the Date/Time utility in the Windows Control Panel or double-click the clock on the Windows **Taskbar**.

---

## Automatically Changing the System Time

Windows 95 and Windows NT will attempt to automatically adjust the clock for daylight savings time. You should turn this feature off by using the Date/Time utility in the Windows Control Panel and use OpenHMI QuickScripts to automatically set the clock at these times.

### ➤ To set the clock forward in the spring:

Create the following Condition QuickScript:

```
$Year == yyyy and $Month == 04 and $Day == dd and
$Hour == 02 and DaylightSavingsTime == 0 ;
```

where: yyyy = the year (i.e., 1993, 1994, or 1995 ...)

dd = the date of the time change

*DaylightSavingsTime* = a user-defined memory discrete tagname to indicate daylight savings time

**ON TRUE:**

```
DaylightSavingsTime = 1;
StartApp "c:\windows\control.exe" ;
SendKeys "%(st)" ;
SendKeys "%(t)" ;
SendKeys "03" ;
SendKeys "~" ;
SendKeys "%({F4})" ;
```

➤ **To set the clock back in the fall:**

Create the following Condition QuickScript:

```
$Year == yyyy and $Month == 10 and $Day == dd and  
$Hour == 02 and DaylightSavingsTime == 1 ;
```

where: yyyy = the year (i.e., 1993, 1994, or 1995 ...)

dd = the day of the time change for that year

*DaylightSavingsTime* = a user-defined memory discrete tagname to  
indicate daylight savings time

---

**Note** Whenever a systems clock is set back, the historical logging engine could overwrite existing data in the historical log file. To prevent this loss of data, we recommend that you first back up the log files before changing the clock back.

---

**ON TRUE:**

```
DaylightSavingsTime = 0 ;  
StartApp "c:\windows\control.exe" ;  
SendKeys "%(st)" ;  
SendKeys "%(t)" ;  
SendKeys "01" ;  
SendKeys "~" ;  
SendKeys "%({F4})" ;
```

## CHAPTER 3

## Tagname Dictionary

The Tagname Dictionary (runtime database) is the heart of OpenHMI. At runtime, the database contains the current value of all of the items in the database. In order to create the runtime database, OpenHMI requires information about all of the variables being created. Each variable must be assigned a tagname and type. OpenHMI also requires additional information for some variable types. For instance, for I/O type tagnames, OpenHMI requires more information in order to be able to acquire the value and convert it for internal use. The Tagname Dictionary is the mechanism used to enter this information.

The two database utility programs, DBDump and DBLoad are also described in this chapter. DBDump allows you to export an OpenHMI application Tagname Dictionary as a text file which can be accessed from another package such as Microsoft Excel for modifying, storing, etc. DBLoad allows a database of tagnames created in another package such as Excel or a DBDump file from another OpenHMI application to be loaded into an existing OpenHMI application.

# Tagname Dictionary Special Features

Beginning with OpenHMI Version 7.0, the Tagname Dictionary has been enhanced to include the following special features:

Feature	Description
<b>Tag Browser</b>	The Tag Browser is used for selecting tagnames and tagname <b>.fields</b> , or any other tag source that supports the Tagname Dictionary interface.
<b>Tagname Cross Referencing</b>	Tagname Cross Referencing allows you to cross reference a tagname to the locations where it is used in your application including windows, scripts, SQL configuration, SPC triggers, and so on. You can print the cross reference information or save it to a file.
<b>Extended Tagname Support</b>	OpenHMI can support up to 1,000 tagnames in its Tagname Dictionary. (The number of tagnames that your system supports is determined by your software license.)

## Tagname Types

When you are defining tagnames in the OpenHMI database, you must assign a specific type to each tagname according to its usage. For example, if the tagname is to read or write values coming to or from another Windows application such as a I/O Server, it must be a I/O type tagname. The following describes each OpenHMI tagname type and its usage.

### Memory Type Tagnames

Memory tagname types exist internally within your OpenHMI application. You use them to create system constants and simulations. You can also use them to create calculated variables that are accessed by other Windows programs. For example, you can define a memory tagname with the initial value of 3.1416 or, you could store recipes in groups of memory tagnames. In simulations, you can use memory tagnames to control the actions of a background QuickScript. For example, you could define a memory tagname "COUNT" what is changed in an action QuickScript to cause various animation effects to occur for the current STEP of a process. There are four Memory types:

#### Memory Discrete

Internal discrete tagname with a value of either 0 (False, Off) or 1 (True, On).

#### Memory Integer

A 32-bit signed integer value between -2,147,483,648 and 2,147,483,647.

#### Memory Real

Floating (decimal) point memory tagname. The floating point value may be between -3.4e38 and 3.4e38. All floating point calculations are performed with 64-bit resolution, but the result is stored in 32-bit.

#### Memory Message

Text string tagname that can be up to 131 characters long.

### I/O Type Tagnames

All tagnames that read or write their values to or from another Windows program are I/O type tagnames. This includes all inputs and outputs from programmable controllers, process computers and data from network nodes. I/O tagnames are accessed either through the Microsoft Dynamic Data Exchange (DDE) or Wonderware SuiteLink communication protocols.

When the value of a read/write I/O type tagname changes, it is immediately written to the remote application. The tagname may also be updated from the remote application whenever the item to which the tagname is linked changes in the remote application. By default, all I/O tagnames are set to Read/Write . However, you can restrict them to read only by selecting the Read Only option in the **Tagname Dictionary** dialog box. There are four I/O types:

#### I/O Discrete

Discrete input/output tagname with a value of either 0 (False, Off) or 1 (True, On).

#### I/O Integer


A 32-bit signed integer value between -2,147,483,648 and 2,147,483,647.

## I/O Real

Floating (decimal) point tagname. The floating point value may be between  $\pm 3.4e^{38}$ . All floating point calculations are performed with 64-bit resolution, but the result is stored in 32-bit.

## I/O Message

Text string input/output tagname that can be up to 131 characters long.

 For more information on using I/O tagnames, see Chapter 9 - I/O Communications.


## Miscellaneous Type Tagnames

There are several special tagname types that you can assign to tagnames to perform complex functions such as creating dynamic alarm displays. There are also indirect tagname types that you can use to reassign a tagname to multiple sources. These special tagname types are described below.

## Group Var

The **Group Var** type is used for a tagname with an assigned Alarm Group to create dynamic alarm displays, disk logs and print logs. You use **Group Var** type tagnames to create alarm windows or alarm logs that display all alarms associated with a specific group variable. You can also control the alarms that are displayed or logged by assigning a different Alarm Group to the **Group Var** tagname.

You can also use a **Group Var** type tagname to create buttons that the operator clicks to selectively display alarms for different areas of a plant in the same alarm window. All of the **.fields** associated with Alarm Groups can be applied to **Group Var** tagnames.

 For more information on Alarms, see Chapter 6 - Alarms/Events.

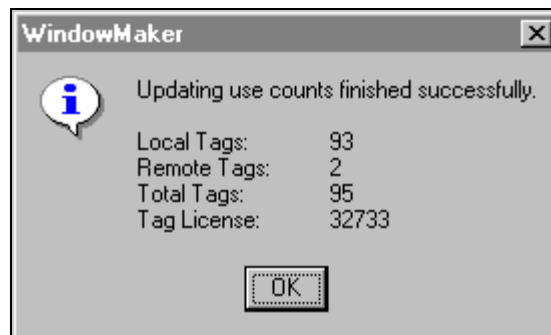


## Extended Tagname Support

OpenHMI can support up to 1,000 tagnames in its Tagname Dictionary. The number of tagnames that your system supports is determined by your software license.

➤ **To determine the tagname support for your system:**

1. Close all your windows.
2. On the **Special** menu, click **Update Use Counts**.
  - 🔊 A message box will appear telling you that updating use counts can take quite a while. You may at that time cancel the command or continue.
3. Click **Yes** to continue updating the use counts.
4. Once the system has completed updating the use counts, the following dialog box appears:



5. The **Tag License** line will display the number of tagnames supported by your license.
6. Click **OK**.

# Defining a New Tagname

Tagnames can be up to 32 characters long and must begin with an alpha character (A-Z or a-z). The remaining characters can be A-Z, a-z, 0-9, !, @, -, ?, #, \$, %, \_, \ and &.

Tagnames are also auto-indexed. For example, if you enter and save tagname R4001, and then click **New**, the tagname will automatically be indexed to R4002. If an tagname contains a character separating numbers, it is auto-indexed by the first whole number OpenHMI finds. For example, N7-0 would be indexed as N7-1. Positive changes only are permitted. For example, R4002 to R4003, R4003 to R4004 and so on.

You need to be careful when you use dashes (-) in tagnames. They are valid for use in tagnames but, they are also used as the negation and subtraction "operator" in expressions or logic. Therefore, some ambiguity arises.

For example, if you use **A=B-C** in an expression, do you mean that **A=B** minus **C** or do you mean that you simply want to assign a tagname named **B-C** to a tagname named **A**? OpenHMI will assume the latter. You can prevent this by separating the tagnames from the operators with blank space(s). For example, **A = B - C**.

Consider this example: **X-101=FT-101\*SP-101**

Can you see where **FT-101** is being multiplied by **SP-101** and assigned to **X-101** due to the fact that no spaces were used?

The first time you access the Tagname Dictionary, the definition for the internal system tagname **\$AccessLevel** is displayed. Once you define tagnames in the Tagname Dictionary, when you access it again, the last edited tagname's definition is displayed.


Click << or >> to browse through the tagname definitions currently stored in your Tagname Dictionary. (The browse buttons will be inactive when there are no previous or next tagnames to display.)

Click **Select** to quickly locate a specific tagname definition. The **Select Tag** dialog box will appear in the selection mode.

 For more information on the Tag Browser, see "The Tag Browser."

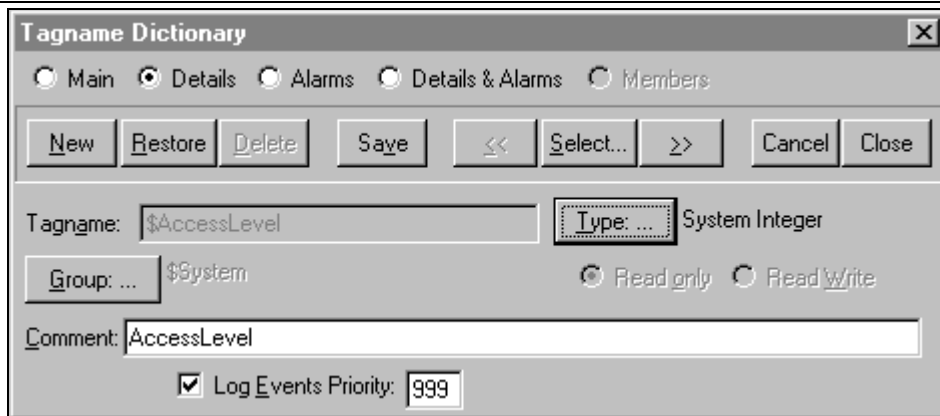
The options at the top of the **Tagname Dictionary** dialog box are used to display the various tagname detail level dialog boxes as follows:

Dialog Box	Description
<b>Main</b>	Displays the main tagname dictionary dialog box (show above). Any changes made to the parent or root tagname can overwrite Member tagname information. After making a change, click <b>Save</b> . A message box will appear asking if you want to overwrite member tagnames with the root tagname changes.
<b>Details</b>	Displays the respective details level dialog box for the tagname type selected.
<b>Alarms</b>	Displays the respective alarms configuration dialog box for the tagname type selected.
<b>Details &amp; Alarms</b>	Displays the both respective details and alarm configuration dialog boxes for the tagname type selected.

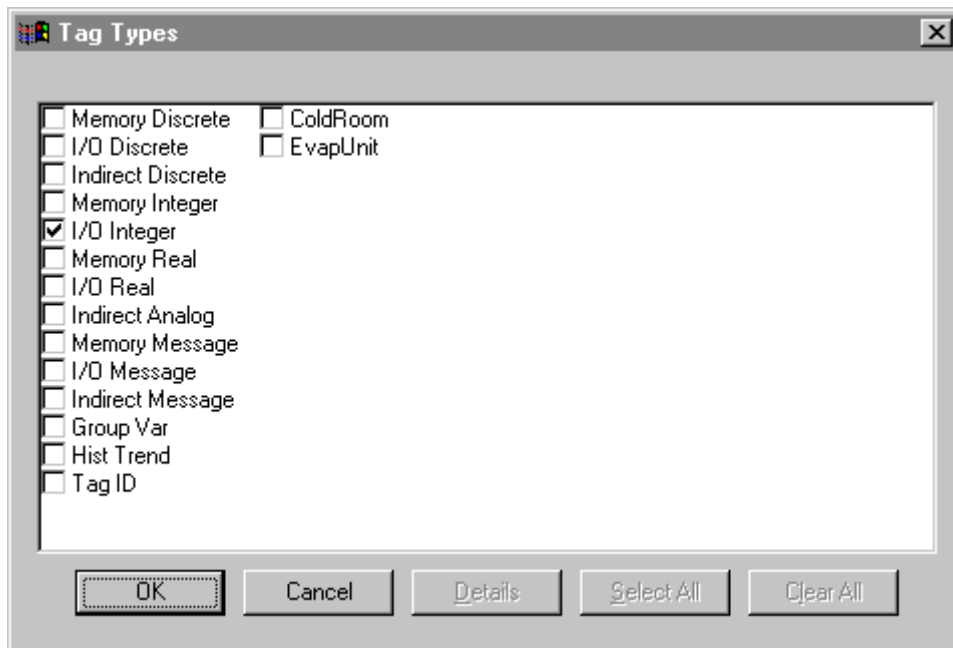
 If you right-click any of the text entry boxes in any of the Tagname Dictionary dialog boxes, a menu will appear displaying the commands that you can apply to the selected text.

➤ **To define a new tagname:**

1. On the **Special** menu, click **Tagname Dictionary** or, in the Application Explorer, double-click **Tagname Dictionary**. The **Tagname Dictionary** dialog box appears:



2. Click **New**. (The **Tagname** box clears.)
3. In the **Tagname** box, type the name you want to use for the new tagname.
  - ☞ Tagnames can be up to 32 characters long and must begin with an alpha character (**A-Z** or **a-z**). The remaining characters can be A-Z, a-z, 0-9, !, @, -, ?, #, \$, %, \_ , \ and &.
4. Click **Type**. The **Tag Types** dialog box appears:



5. Select the type for the tagname, then click **OK**. The respective details dialog box for the selected type will appear. (The detail dialog boxes are described later in this chapter.)

☞ For more information on tagname types, see "Tagname Types."

---

**Note** If a tagname is currently linked to an object or used in a QuickScript, its type can only be changed when WindowViewer is not running.

---

6. Click **Group** to assign the tagname to a specific Alarm Group. The **Alarm Groups** dialog box will appear. Select the name of the Alarm Group that you want to assign to the tagname, then click **Done**.

---

**Note** If you do not assign the tagname to a specific Alarm Group, by default, OpenHMI will assign it to the root group, **\$System**.

---

---

Once you create a tagname and assign it to an Alarm Group, if you do not close the dialog box, all subsequent tagnames that you define will be assigned to the same Alarm Group, unless you change it.

---

For more information on defining Alarm Groups, see Chapter 6 - Alarms/Events.

7. For I/O type tagnames, select **Read Only** to restrict the tagname to read only capabilities in runtime.
8. For I/O type tagnames, select **Read Write** to grant the tagname read and write capabilities in runtime.
9. In the **Comment** box, type any miscellaneous comment you want the system to store regarding your tagname (up to 50 characters). You can configure your alarm windows to display the these comments whenever the tagname is in alarm.
  - ☞ The first time you access the **Tagname Dictionary** dialog box, the default comment for the internal system tagname **\$AccessLevel** will be displayed in the **Comment** box. You should delete this comment to prevent it from being associated with any tagnames that you define. To delete the comment, select it and press the DEL key.
11. Select **Log Events** if you want to log all data value changes to the tagname that are initiated by the operator, I/O, a QuickScript or by the system.
  - ☞ When you define a tagname to do event monitoring, an event message is logged to the alarm system each time the tagname's value changes. The event message logs how the value changed, whether the change was initiated by the operator, I/O, scripts or the system.

When you select **Log Events**, the **Priority** field becomes active. The value you type for the **Priority** determines the event priority level for the tagname. Valid entries in this field are 1 to 999 where, 1 is the highest and 999 is the lowest priority.
  - For more information on events and priorities, see Chapter 6 - Alarms/Events.
12. Select **Retentive Value** if you want to retain the current value of the tagname whenever WindowViewer is exited. This value will be used as the initial value for the tagname whenever WindowViewer is restarted.
  - ☞ Retentive values cannot be selected or cleared for new or existing tagnames if WindowViewer is running. When you select this option, the initial value of the tagname will constantly be updated to reflect the current value of the tagname. When WindowViewer is exited, the initial value is set based on the last retained value. If this option is later cleared, the initial value of the tagname will be set to the last retained value.
13. Select **Retentive Parameters** if you want to retain any changes the operator makes to the value of any alarm limit fields for the tagname. This value will be used as the initial value for the alarms when WindowViewer is restarted.

---

**Note** Since changes are logged immediately, we strongly recommended that you only select the above two retentive options for values that do not change often.

---
14. Define the details for the type of tagname.
15. Click **Done**.

## Defining Tagname Details

The initially displayed **Tagname Dictionary** dialog box is used to input basic tagname information. Many points, especially inputs and outputs, require greater detail to be properly handled. For each type of tagname specified, a specific details dialog box exists that you use to define the details for the tagname type.

Most of the tagname types have their own specific detail level dialog boxes and alarm condition dialog boxes. By default, when you select the type for your tagname, its respective details level dialog box will appear.

Once you have completed defining the basic tagname, you will need to define the details for the tagname and, if required, the alarm conditions. The steps that you need to follow to define the details for each tagname type are described in the following sections.

### Defining Memory Discrete Tagname Details

Memory discrete type tagnames exist internally within your OpenHMI application. You define a **Memory Discrete** type tagname when you need an internal tagname with a value of either 0 (False, Off) or 1 (True, On).

➤ **To define the details for a memory discrete tagname:**

1. When you select **Memory Discrete** as the type for your tagname, the following details dialog box will appear.

☞ If it does not appear, click **Details** at the top of the **Tagname Dictionary** dialog box.

2. Click the **Initial Value** that you want stored in the tagname when the runtime database is first loaded.
3. If you define a discrete alarm state for this tagname that is "on" when the tagname's value is equal to 1 (On, True), type the message in the **On Msg** box that you want to be displayed in your alarm window's value/limit field.
4. If you define a discrete alarm state for this tagname that is "on" when the tagname's value is equal to 0 (Off, False), type the message in the **Off Msg** box that you want to be displayed in your alarm window's value/limit field.
5. If you want to define alarm conditions for the tagname, click either **Alarms** or **Details & Alarms** at the top of the **Tagname Dictionary** dialog to display the respective alarm conditions dialog box for the type of tagname you are defining.
 

☞ For more information on alarms, see Chapter 6 - Alarms/Events.
6. Once you have completed defining your tagname, click **Done** to save your tagname definition and close the tagname dialog boxes.

### Defining Memory Analog Tagname Details

Memory analog type tagnames exist internally within your OpenHMI application. There are two memory analog types: **Memory Integer** and **Memory Real**. You define a **Memory Integer** type tagname when you need an internal tagname with a 32-bit signed integer value between -2,147,483,648 and 2,147,483,647.

You define a **Memory Real** type tagname when you need an internal tagname with a floating point value between  $-3.4e^{38}$  and  $3.4e^{38}$ . (All floating point calculations are performed with 64-bit resolution, but the result is stored in 32-bit.)

➤ **To define the details for a memory analog tagname:**

1. When you select **Memory Integer** or **Memory Real** as the type for your tagname, the following details dialog box will appear.

☞ If it does not appear, click **Details** at the top of the **Tagname Dictionary** dialog box.

Initial Value:	<input type="text" value="0"/>	Eng Units:	<input type="text"/>
Min Value:	<input type="text" value="0"/>	Deadband:	<input type="text" value="0"/>
Max Value:	<input type="text" value="9999"/>	Log Deadband:	<input type="text" value="0"/>

2. In the **Initial Value** box, type the value you want stored in the tagname when the runtime database is first loaded.
3. In the **Min Value** box, type the minimum value you want to use for I/O and the **.Min EU** tagname .field.
4. In the **Max Value** box, type the maximum value you want to use for I/O and the **.Max EU** tagname .field.
5. In the **Eng Units** box, enter the label you want to use for the tagname's engineering units.
6. In the **Deadband** box, type the amount the tagname's engineering units can change before the database is updated.
7. If you want to define alarm conditions for the tagname, click either **Alarms** or **Details & Alarms** at the top of the **Tagname Dictionary** dialog to display the respective alarm conditions dialog box for the tagname type you are defining.
 

☞ For more information on alarm conditions, see "Defining Tagname Alarm Conditions."
8. Once you have completed defining your tagname, click **Close** to save your tagname definition and close the tagname dialog boxes.

## Defining Memory Message Tagname Details

Memory message type tagnames exist internally within your OpenHMI application. You define a **Memory Message** type tagname when you need an internal text string tagname that can be up to 131 characters long.

➤ **To define the details for a memory message tagname:**

1. When you select **Memory Message** as the type for your tagname, the following details dialog box will appear.

☞ If it does not appear, click **Details** at the top of the **Tagname Dictionary** dialog box.

Maximum Length:	<input type="text" value="131"/>
Initial Value:	<input type="text"/>

2. In the **Maximum Length** box, type the maximum number of characters to be allowed in the tagname's message. (OpenHMI allows a maximum of 131, which is displayed as the default.)
3. In the **Initial Value** box, type the text string that you want displayed for the tagname when WindowViewer is initially started.
4. Once you have completed defining your tagname, click **Close** to save your tagname definition and close the tagname dialog boxes.

## Defining I/O Discrete Tagname Details

All tagnames that read or write their values to or from another Windows program are I/O type tagnames. This includes all inputs and outputs from programmable controllers, process computers, other Windows programs and data from network nodes.

You define an **I/O Discrete** type tagname when you need an I/O tagname with a value of either 0 (False, Off) or 1 (True, On).

➤ **To define the details for a I/O discrete tagname:**

1. When you select **I/O Discrete** as the type for your tagname, the following details dialog box will appear.
  - ☞ If it does not appear, click **Details** at the top of the **Tagname Dictionary** dialog box.

2. Click the **Initial Value** that you want stored in the tagname when the runtime database is first loaded. (**Off** equals **0**, **On** equals **1**.)
3. Click the **Input Conversion** that you want applied to the value when the runtime database is updated:
  - Direct** - The I/O input value is read unchanged directly from the server program.
  - Reverse** - The I/O input value is reversed when read from the server program. For example, if the I/O input value in the server program is 0, OpenHMI will automatically reverse it, save it and display a 1.
4. If you define a discrete alarm state for this tagname that is "on" when the tagname's value is equal to 1 (On, True), type the message in the **On Msg** box that you want to be displayed in your alarm window's value/limit field.
5. If you define a discrete alarm state for this tagname that is "on" when the tagname's value is equal to 0 (Off, False), type the message in the **Off Msg** box that you want to be displayed in your alarm window's value/limit field.
6. Click **Access Name** to define or select the Access Name that you want to assign to this tagname. (If an Access Name already appears to the right of this button, and you do not define or select a different one, it will be assigned to the tagname.
  - ☞ For more information on Access Names, see Chapter 8 - I/O Communications.
7. In the **Item** box, type the valid item name for the data point in the server program that the tagname will read/write its value to/from. For example, if you want to read

a value from a register in a PLC, enter the valid identification for that register as the item name.

☞ Item names are auto-indexed. For example, if you enter and save item name R4001, then click **New** (to define a new tagname), the item name will automatically be indexed to R4002. If an item name contains a character separating numbers, it is auto-indexed by the first whole number OpenHMI finds. For example, N7-0 would be indexed as N7-1. Positive changes only are permitted. For example, R4002 to R4003, R4003 to R4004 and so on.

8. Select the **Use Tagname as Item Name** option if you want to use the tagname for the item name.

9. If you want to define alarm conditions for the tagname, click either **Alarms** or **Details & Alarms** at the top of the **Tagname Dictionary** dialog to display the respective alarm conditions dialog box for the tagname type you are defining.

☞ For more information on alarm conditions, see "Defining Tagname Alarm Conditions."

10. Once you have completed defining your tagname, click **Close** to save your tagname definition and close the tagname dialog boxes.

## Defining I/O Analog Tagname Details

All tagnames that read or write their values to or from another Windows program are I/O type tagnames. This includes all inputs and outputs from programmable controllers, process computers, other Windows programs and data from network nodes. There are two memory analog types: **I/O Integer** and **I/O Real**.

You define an **I/O Integer** type tagname when you need an I/O tagname with a 32-bit signed integer value between -2,147,483,648 and 2,147,483,647.

You define an **I/O Real** type tagname when you need an I/O tagname with a floating point value between  $-3.4e^{38}$  and  $3.4e^{38}$ . (All floating point calculations are performed with 64-bit resolution, but the result is stored in 32-bit.)

### ➤ To define the details for an I/O analog tagname:

1. When you select **I/O Integer** or **I/O Real** as the type for your tagname, the following details dialog box will appear.

☞ If it does not appear, click **Details** at the top of the **Tagname Dictionary** dialog box.

The screenshot shows a dialog box titled "I/O Analog Tagname Details". It has the following fields and controls:

- Initial Value:** Input field with value 0.
- Min EU:** Input field with value 0.
- Max EU:** Input field with value 9999.
- Deadband:** Input field with value 0.
- Min Raw:** Input field with value 0.
- Max Raw:** Input field with value 9999.
- Eng Units:** Input field (empty).
- Access Name:** Button labeled "Access Name: ..." followed by the text "Assigned".
- Conversion:** Radio button group with "Linear" selected and "Square Root" unselected.
- Item:** Input field (empty).
- Use Tagname as Item Name:** Unchecked checkbox.
- Log Deadband:** Input field with value 0.

2. In the **Initial Value** box, type the value you want stored in the tagname when the runtime database is first loaded.

3. In the **Deadband** box, type the amount the engineering units for the tagname can change before the database is updated.



4. In the **Min EU** box, type the engineering units value for the tagname when the minimum raw count value is received.
5. In the **Min Raw** box, type the minimum value of the low clamp on the raw I/O integer values.
6. In the **Max EU** box, type the engineering units value for the tagname when the maximum raw count value is received.
7. In the **Max Raw** box, type the maximum value of the high clamp on the raw I/O integer values.

☞ You can use the **Min EU**, **Min Raw**, **Max EU** and **Max Raw** values to scale your I/O tagnames.

☞ For more information on scaling tagnames, see "Scaling I/O Tagnames."

8. In the **Eng Units** box, enter the label you want to use for your tagname's engineering units.
9. Select the type of **Conversion** that you want the database to use to scale the raw counts when calculating the engineering units.

**Linear** - The result is calculated using linear interpolation between the end points.

The algorithm for linear scaling of input is:

$$EUValue = (RawValue - MinRaw) * ((MaxEU - MinEU) / (MaxRaw - MinRaw)) + MinEU$$

The algorithm for linear scaling of output is:

$$RawValue = (EUValue - MinEU) * ((MaxRaw - MinRaw) / (MaxEU - MinEU)) + MinRaw$$

**Square Root** - The raw counts values are used for interpolation. This is useful for scaling inputs from nonlinear devices such as pressure transducers.

The algorithm for square root scaling of input is:

$$EUValue = \text{sqrt}(RawValue - MinRaw) * ((MaxEU - MinEU) / \text{sqrt}(MaxRaw - MinRaw)) + MinEU$$

The algorithm for square root scaling of output is:

$$RawValue = \text{square}((EUValue - MinEU) * (\text{sqrt}(MaxRaw - MinRaw) / (MaxEU - MinEU))) + MinRaw$$

10. Click **Access Name** to define or select the Access Name that you want to assign to this tagname. (If an Access Name already appears to the right of this button, and you do not define or select a different one, it will be assigned to the tagname.)

☞ For more information on Access Names, see Chapter 8 - I/O Communications.

11. In the **Item** box, type the valid item name for the data point in the server program that the tagname will read/write its value to/from. For example, if you want to read a value from a register in a PLC, enter the valid identification for that register as the item name.

☞ Item names are auto-indexed. For example, if you enter and save item name R4001, then click **New** (to define a new tagname), the item name will automatically be indexed to R4002. If an item name contains a character separating numbers, it is auto-indexed by the first whole number OpenHMI finds. For example, N7-0 would be indexed as N7-1. Positive changes only are permitted. For example, R4002 to R4003, R4003 to R4004 and so on.

12. Select the **Use Tagname as Item Name** option if you want to use the tagname for the item name.
13. If you want to define alarm conditions for the tagname, click either **Alarms** or **Details & Alarms** at the top of the **Tagname Dictionary** dialog to display the respective alarm conditions dialog box for the tagname type you are defining.

☞ For more information on alarm conditions, see "Defining Tagname Alarm Conditions."

- Once you have completed defining your tagname, click **Close** to save your tagname definition and close the tagname dialog boxes.

## Defining I/O Message Tagname Details

All tagnames that read or write their values to or from another Windows program are I/O type tagnames. This includes all inputs and outputs from programmable controllers, process computers, other Windows programs and data from network nodes.

You define an **I/O Message** type tagname when you need an internal text string tagname that can be up to 131 characters long.

➤ **To define the details for a I/O message tagname:**

- When you select **I/O Message** as the type for your tagname, the following details dialog box will appear.

☞ If it does not appear, click **Details** at the top of the **Tagname Dictionary** dialog box.

The screenshot shows a dialog box with the following elements:

- Maximum Length:** A text box containing the number "131".
- Initial Value:** An empty text box.
- Access Name:** A button labeled "Access Name: ..." followed by the text "Unassigned".
- Item:** An empty text box.
- Use Tagname as Item Name:** A checkbox that is currently unchecked.

- In the **Maximum Length** box, type the maximum number of characters to be allowed in the tagname's message. (OpenHMI allows a maximum of 131, which is displayed as the default.)
- In the **Initial Value** box, type the text string that you want displayed for the tagname when WindowViewer is initially started.
- Click **Access Name** to define or select the Access Name that you want to assign to this tagname. (If an Access Name already appears to the right of this button, and you do not define or select a different one, it will be assigned to the tagname.)

☞ For more information on Access Names, see Chapter 8 - I/O Communications.

- In the **Item** box, type the valid item name for the data point in the server program that the tagname will read/write its value to/from. For example, if you want to read a value from a register in a PLC, enter the valid identification for that register as the item name.

☞ Item names are auto-indexed. For example, if you enter and save item name R4001, then click **New** (to define a new tagname), the item name will automatically be indexed to R4002. If an item name contains a character separating numbers, it is auto-indexed by the first whole number OpenHMI finds. For example, N7-0 would be indexed as N7-1. Positive changes only are permitted. For example, R4002 to R4003, R4003 to R4004 and so on.

- Select the **Use Tagname as Item Name** option if you want to use the tagname for the item name.
- Once you have completed defining your tagname, click **Close** to save your tagname definition and close the tagname dialog boxes.

## Defining Tagname Alarm Conditions

You can define alarm conditions for tagnames at the same time that you define the tagname. There are two types alarm detail dialog boxes. One for discrete type tagnames and one for analog (integer or real) type tagnames.

### Defining Discrete Tagname Alarm Conditions

You can define an alarm condition for a discrete type tagname's **On** state or **Off** state.

➤ **To define alarm conditions for a discrete tagname:**

1. On the **Special** menu, click **Tagname Dictionary** or, in the Application Explorer, double-click **Tagname Dictionary**. The **Tagname Dictionary** dialog box will appear.
2. Click either **Alarms** or **Details & Alarms** at the top of the **Tagname Dictionary** dialog box to display the discrete alarm details dialog box:

3. Click the **Alarm State** that you want the tagname to be in when in alarm.
4. In the **Priority** box, type a value between 1 and 999 (1 is the highest priority and 999 is the lowest). You can use this priority value to select the alarms that you want to be displayed in a window, logged to disk or printed.
5. Click **Close** (in the **Tagname Dictionary** dialog box) to save your tagname definition and close the tagname dialog boxes.

### Defining Analog Tagname Alarm Conditions

➤ **To define alarm conditions for an analog tagname:**

1. On the **Special** menu, click **Tagname Dictionary** or, in the Application Explorer, double-click **Tagname Dictionary**. The **Tagname Dictionary** dialog box will appear.
2. Click either **Alarms** or **Details & Alarms** at the top of the **Tagname Dictionary** dialog box to display the analog alarm details dialog box:

3. Select the alarm types (**LoLo**, **Low**, **High**, **HiHi**) that you want to use to detect when the value of an analog type tagname is beyond an absolute limit.
4. In the **Alarm Value** box, type the limit value for the alarm.

For example, in the case of **LoLo** and **Low** alarms, an alarm condition exists whenever the value of the tagname is less than the **Alarm Value**. In the case of

**High** and **HiHi** alarms, an alarm condition exists whenever the value of the tagname is greater than the **Alarm Value**. These fields support the use of real numbers (i.e., 100.75).

5. In any of the **Pri** (priority) boxes type a number between 1 and 999 (1 is the highest priority and 999 is the lowest). You can use the priority value to select the alarms you want to be displayed in a window, logged to disk or printed.
6. In the **Value Deadband** box, type the number of engineering units the tagname value must drop below the alarm value before it is taken out of alarm.  
  
For example, to return-to-normal from an alarm condition, a tagname value must not only return inside its alarm limit, but also return through your specified **Value Deadband**. The **Value Deadband** prevents "nuisance" alarms caused by repetitive re-annunciation of an alarm (where the tagname value 'hovers' around the limit, continually hopping in and out of alarm).
7. Select the deviation (**Minor** and **Major Deviation**) alarm types you want to use to detect when the value of an analog type tagname is in a major or minor deviation from the specified **Target** value.
8. In the **%Deviation** box, type the percentage that the analog tagname can deviate from the **Target** value to produce a minor or major deviation alarm condition. It is expressed as a percentage of the range of the tagname. The range is defined by the **Min EU** and **Max EU** values entered in the tagname's details dialog box.
9. In the **Target** box, type the desired or reference value of the tagname from which minor and/or major deviation percentages are based.
10. In the **Deviation Deadband %** box, type the deviation percentage the tagname value must drop below the target before it is taken out of alarm. For example, let's assume the following setup for an integer tagname:

Minimum Value = -1000

Maximum Value = 1000


Minor Deviation % = 10

Major Deviation % = 15

Target = 500

To calculate at what value the Minor or Major Deviation alarm will take place if the total range of the tagname is **-1000 to +1000 or 2000**, multiply **2000** by either the Minor or Major Deviation percentage (**2000 x .10 (Minor) =200**). If the Target is **500**, a Minor Deviation will occur whenever the tagname's value drops below **300** or rises above 700.

11. Select **Rate of Change** if you want to detect when the value of an alarm changes an excessive amount for a specified time interval. The tagname is tested for a **Rate of Change** alarm whenever its value changes. At this time, the change rate is calculated using the previous value, the time of the last update, the current value, and the current time. This is compared to the rate of change allowed in the alarm definition. If the rate of change is greater than the alarm limit, the **Rate of Change** alarm condition is set for the tagname. A **Rate of Change** alarm remains in effect until the next change in the tagname is less than the excessive change amount for the time interval.
12. In the **% per** box, type the maximum allowable percentage change.
13. Select **Sec**, **Min**, or **Hr** for the time interval units of the change.

 For more information on alarms, see Chapter 6 - Alarms/Events.

# The Tag Browser

The Tag Browser is your primary tool for viewing and selecting tagnames and tagname **.fields** from OpenHMI applications, or any other tag source that supports the OpenHMI Tagname Dictionary interface. It allows you to select existing tagnames, add new tagnames and view basic Tagname Dictionary information. You also use the Tag Browser to access the dialog boxes that allow you to perform tagname editing and replication.

The first time you access the Tag Browser, by default, **<local>** will be selected for the tag source. Meaning that the tagnames in the local application's Tagname Dictionary will be displayed. Thereafter, the last accessed tag source's tagnames will be displayed.

The Tag Browser operates in two modes; "Filtered Selection Mode" and "Unlimited Selection Mode." The mode for the Tag Browser is determined by the method you use to access it. The following lists the primary methods that you can use to access the Tag Browser in each mode:

## Unlimited Selection Mode

- Double-clicking an animation link tagname or expression input box.
- Double-clicking an ActiveX or wizard tagname or expression input box.
- Double-clicking a blank area in any OpenHMI QuickScript window.
- In the OpenHMI QuickScript editor, selecting the **Tagname** command on the **Insert** menu.
- Pressing the ALT + N keys in the OpenHMI QuickScript editor.
- Double-clicking a blank **New Name** box in the **Substitute Tagnames** dialog box.
- Double-clicking the **Tagname.FieldName** input box in the SQL Access **Bind List Configuration** dialog box.

## Filtered Selection Mode

- Clicking the **Select** button in the Tagname Dictionary.
- When WindowMaker is running, double-clicking a cell in the **Unit#** column in a Recipe Manager **Unit Template** definition.

The Tag Browser's status bar provides status on the following items for the currently displayed tag source:

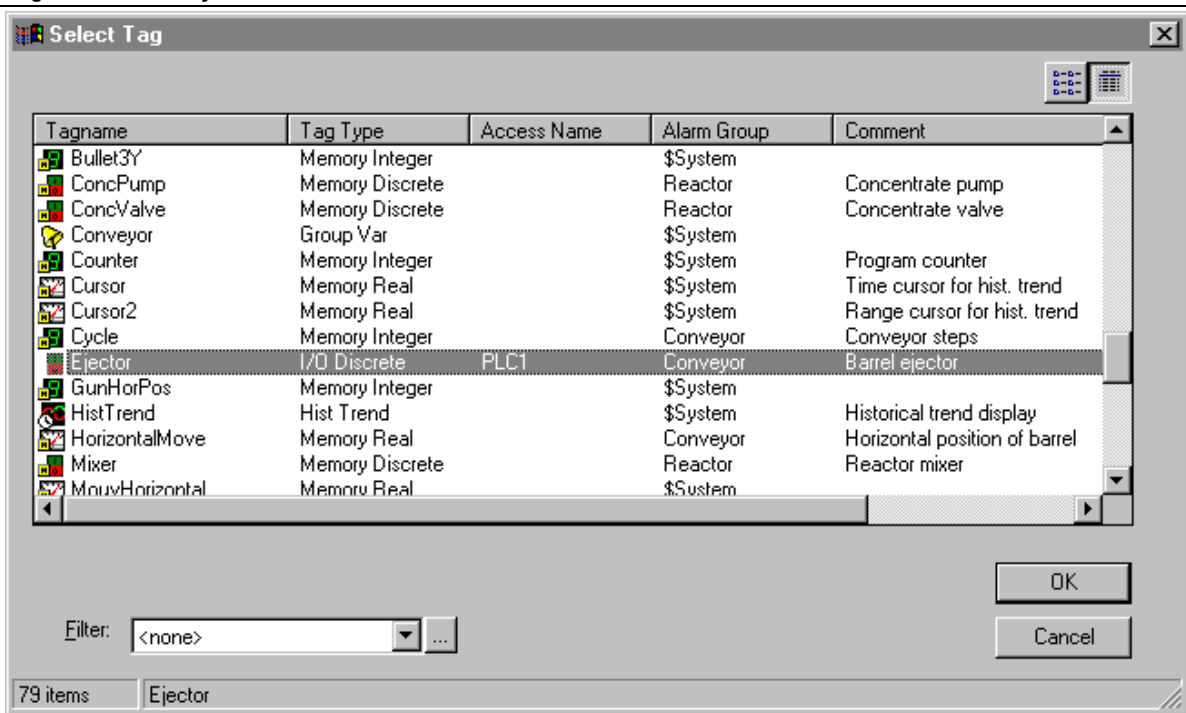
- Total number of items in the application.
- The name of the currently selected item.
- Tagname **.field** selected, if any.
- The Access Name associated with the tag source.

## Tag Browser Selection Modes

The Tag Browser operates in two selection modes; Filtered Selection Mode and Unlimited Selection Mode.

### Filtered Selection Mode

If you click **Select** in the **Tagname Dictionary** dialog box, the tagnames displayed (and available for selecting) will be limited to the current OpenHMI application. For example:



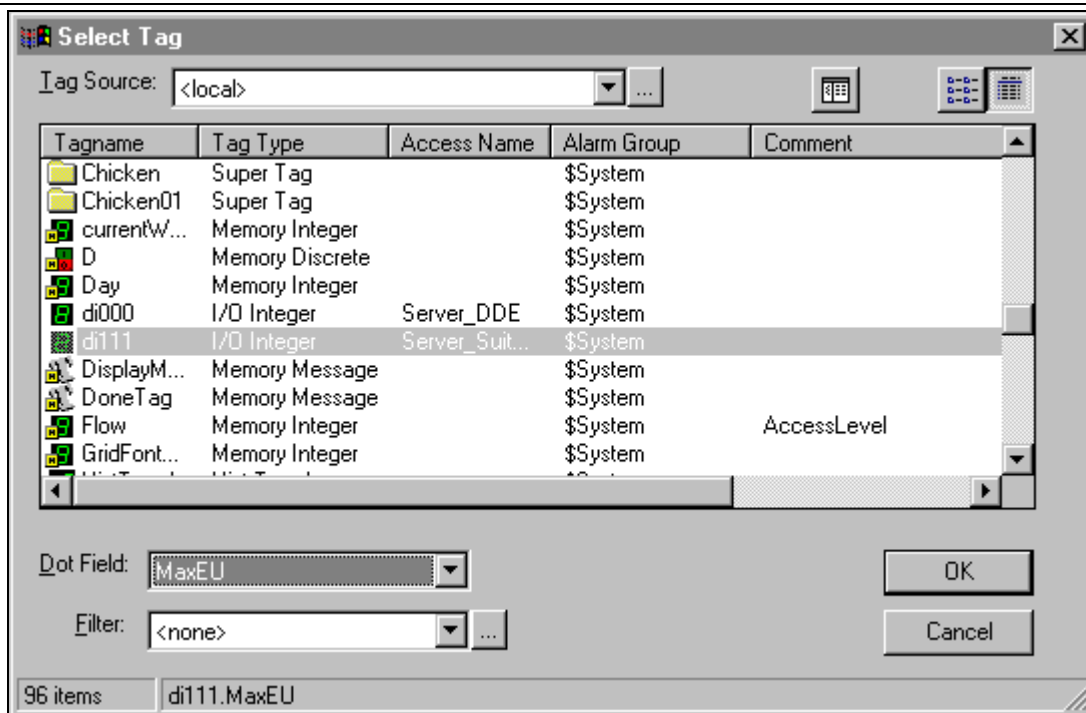
When you access the Tag Browser from the Tagname Dictionary and you select a tagname in this view, it's Tagname Dictionary definition will appear after you click **OK**.

**Note** Tagname **.fields** cannot be selected in this mode.

## Unlimited Selection Mode

The unlimited selection mode is accessed by double-clicking in a blank area in any OpenHMI QuickScript window, animation link tagname or expression box or, a blank **New Name** box in the **Substitute Tagnames** dialog box. The tagnames defined can be displayed and selected in this mode.

Tagname **.fields** can also be selected for the tagname in this mode. When you select a tagname and/or tagname.**field** in this mode, it is automatically entered into the OpenHMI QuickScript, animation link tagname or expression box or, other location from which you accessed the Tag Browser. For example:



➤ **To select a .field:**

1. Click the **Dot Field** arrow to open the list of **.fields** that you can associate with the type of tagname currently selected.
  - ☞ By default, **<none>** will initially be displayed for all types of tagnames.

---

**Note** **Dot Field** is not available when you access the Tag Browser from the Tagname Dictionary.

---

2. Click the **.field** in the list that you want to append to the selected tagname.
  - ☞ Not every tagname type has the same **.fields**. For example, a **Discrete** type tagname has **.OnMessage**, whereas an analog does not. If you select a **Discrete** type tagname and you assign **.OnMessage** to it, and then you select another **Discrete** type tagname, the displayed **.field** list will not change. But, if you select another type of tagname in the control view list, for example an analog, the displayed **.field** will revert to **<none>**.

## Tag Browser Views

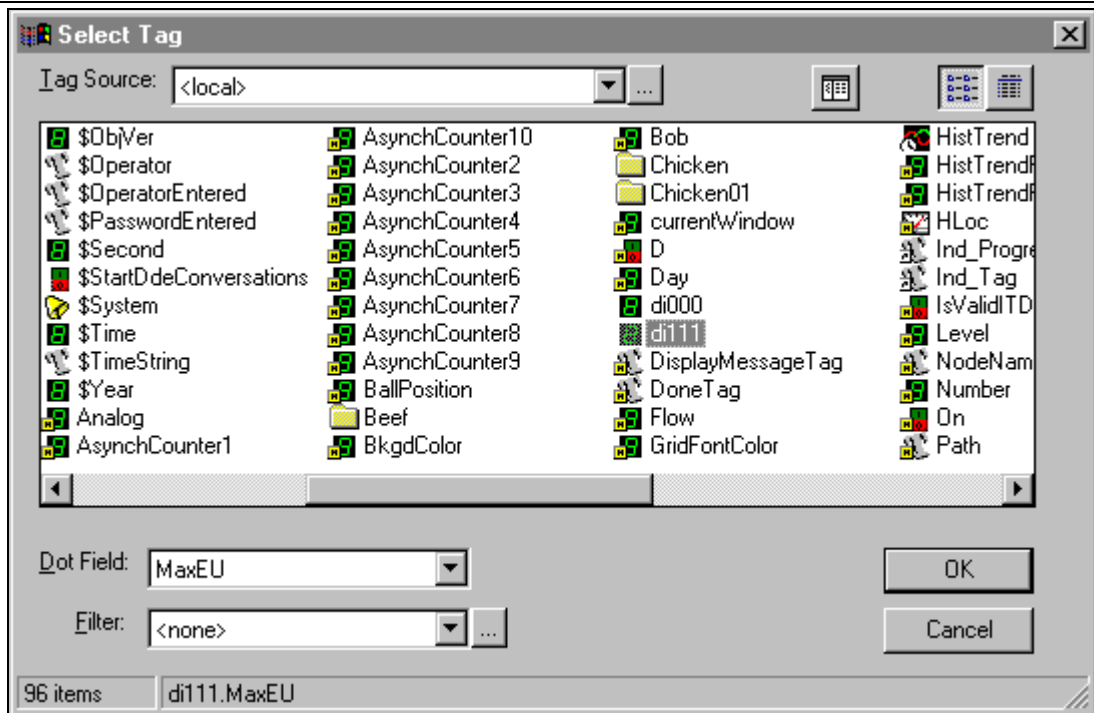
The Tag Browser supports three control views; Tagname List Control, Tagname Details Control and Tagname Tree View Control.



### List View

The list view is used to display and select tags within the current selection mode (described above). The Tagname List Control view displays the tagnames in two views depending upon the state of the **List View** and **Details View** buttons:

When you select list view, small icons will be displayed next to the tagnames with icons displayed according to the type of each tagname. No other fields will be displayed in the list view. For example:

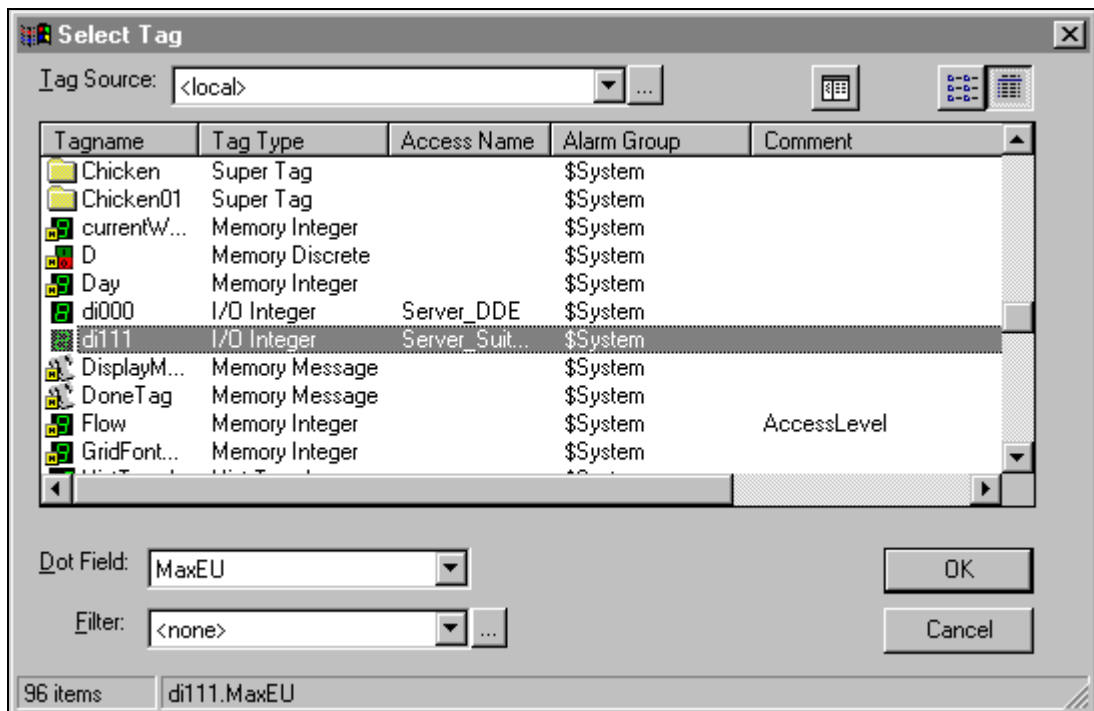


To refresh the display in the **Select Tag** dialog box, press F5.



## Details View

When you select details view, the tagnames and their details are displayed in a multi-column format. The details displayed are Tagname Name, Tagname Type, Access Name, Alarm Group and Comment. You can sort the list by each detail type by clicking on its column header name. An item can be selected by clicking on any portion of its display, not just the tagname. (The entire row will be highlighted.) For example:



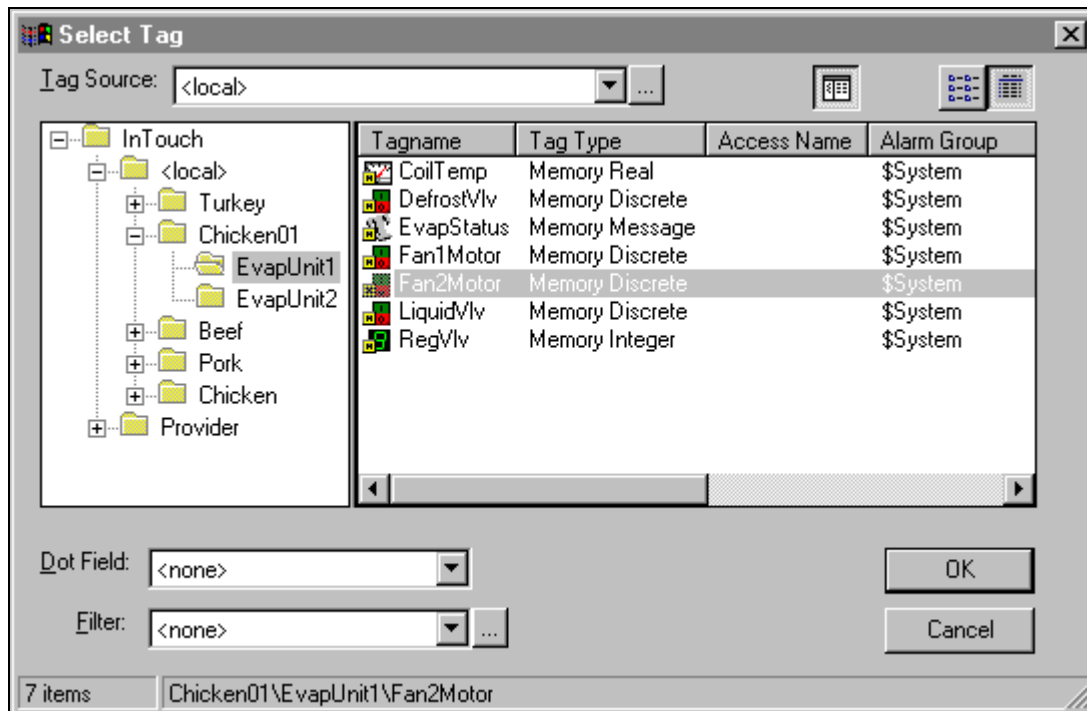
☞ When you switch views, the selected tagname will remain visible and highlighted in the new view.



## Tree View

The Tree View displays the tagnames in two views depending upon the state of the **List View** and **Details View** buttons. When you select the tree view, a pane appears on the left side of the dialog box.

If the **Details View** mode is active when you select the Tree View, Tag Browser appear as follows:



☞ To expand a listing in the Tree View, double-click the application name or, click the **+**. To collapse a listing, double-click the application name again or, click the **-**. Double-clicking an application in the tree view pane is the same as selecting it in the **Tag Source** list.

---

**Note** When you "drill down" through different levels in the Tag Browser, you can use the BACKSPACE key to "back up" to the previous level.

---

## Defining Tag Sources

Adding, deleting, or editing sources are not supported by OpenHMI.

## Defining Tag Browser Filters

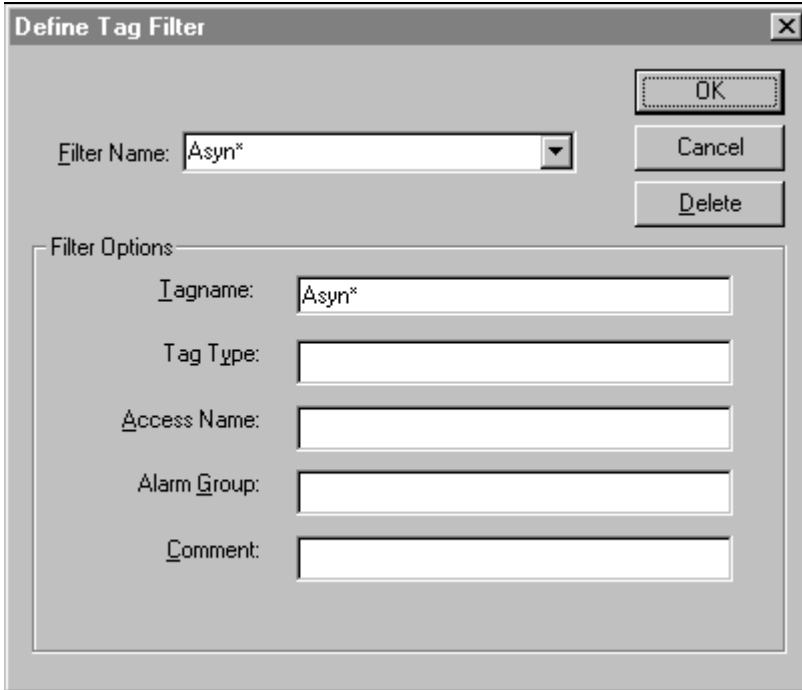
You will use the procedures described in this section to define the filters (search criteria) you want to use to populate the Tag Browser. By creating filters, you can sort any tagname list and display only the tagnames that meet the criteria you specify. You can sort the tagnames based on **Tagname**, **Tag Type**, **Access Name**, **Alarm Groups** and tagname **Comments**. You can use one or a combination of any of these items to set the criteria for your display. You can also save each filter instance and reuse it at any time.

☞ For example, if you have 40,000 tagnames defined in your Tagname Dictionary and you only need to deal with the 20 or so that are assigned to a particular Access Name or Alarm Group, you can create a filter and specify the Access Name and/or

Alarm Group as the criteria that the tagnames must meet in order to be displayed in the Tag Browser.

➤  **To define a search filter:**

1. Click the Define Filter button. The **Define Tag Filter** dialog box appears:



☞ If you right click the mouse in any of the text entry boxes, a menu will appear displaying the commands that you can apply to the selected text.

2. In the **Filter Name** box, type a unique name to identify the filter that you are defining or, click the **Filter Name** arrow to select a previously defined filter name from the list. (As you define filters, the **Filter Name** you type is added to the list.)

☞ All of the **Filter Option** controls (**Tagname**, **Tag Type**, **Access Name**, **Alarm Group** and **Comment**) allow you to enter a wildcard expression to limit the scope of your search. If no filter is used, all of the tagnames in the currently displayed tag source will be displayed.

The multiple wildcard is the asterisk (\*). For example, "Asyn\*" would search for all tagnames beginning with the character "Asyn".

The single character wildcard is the tilde (?). For example, the filter, "Tag?" would search for all four character tagnames that begin with "Tag". The filter, "Tag??", would search for all five character tagnames that begin with "Tag", and so on.

Any sequence of valid OpenHMI tagname characters, together with the two wildcard characters, is acceptable in a filter. The valid tagname characters are: A-Z, a-z, 0-9, !, @, -, ?, #, \$, %, \_, \ and &.

3. In the **Tagname** box, type the tagname expression that you want to use as a filter. If left blank, the system will ignore this field in the filter definition.
4. In the **Access Name** box, type the local Access Name expression that you want to use as a filter. If left blank, the system will ignore this field in the filter definition.
5. In the **Alarm Group** box, type the name of the Alarm Group expression that you want to use as a filter. If left blank, the system will ignore this field in the filter definition.

6. In the **Comment** box, type the comment expression you want to use as a filter. If left blank, the system will ignore this field in the filter definition.
  7. Click **OK** to close dialog box.
    - ☞ The **Filter Name** will now appear in the **Filter** list in the Tag Browser and you can select it to display only the tagnames meeting the criteria specified in the filter.
- **To delete a search filter:**
1. Click the **Filter** arrow and select the filter name in the list that you want to delete.
  2. Click **Delete**. The filter is immediately deleted.

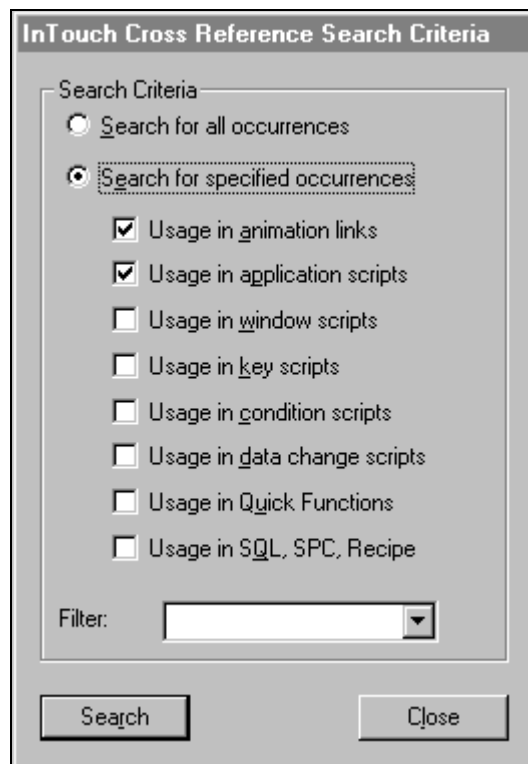
# OpenHMI Cross Reference Utility

The Tagname Cross Referencing utility allows you to determine your tagname usage in animation links, wizards, OpenHMI QuickScripts, ActiveX controls, scripts and the following OpenHMI add-on programs, SPC Pro, SQL Access Manager and Recipe Manager. For all objects such as wizards, ActiveX controls and animation links, it displays the window name and the coordinates of all objects linked to the tagname. It also allows you to view any QuickScript where a tagname is found.

☞ For convenience, the Tagname Cross Reference utility can remain open in WindowMaker while you perform other tasks.

➤ **To use the OpenHMI Cross Reference utility:**

1. On the **Special** menu, click **Cross Reference** or, in the Application Explorer double-click **Cross Reference**. The **OpenHMI Cross Reference Search Criteria** dialog box appears:



2. The **Search Criteria** group allows you to limit the scope of your search. You can easily determine the scope by selecting only the options required.

**Search for all occurrences**

Search for all uses of the tagname in animation links, OpenHMI QuickScripts and all add-on programs such as SPC, SQL Access Manager, Recipe Manager, and so on.

**Search for specific occurrences**

Search for only the tagname only in the specified options. For example, if you only want to search for the usage in window scripts, only select **Usage in window scripts**.

3. In the **Filter** box, type a unique name to identify the filter that you are defining or, click the **Filter** arrow to select a previously defined filter from the list. (As you define filters, the name you type is added to the **Filter** list.)

☞ The filter editor control allows you to enter a wildcard expression to limit the scope of the tagnames in your search. If no filter is used, the information for all tagnames in the current application will be acquired.

The multiple wildcard is the asterisk symbol (\*). For example, "Asyn\*" would search for all tagnames beginning with the characters "Asyn".

The single character wildcard is the question mark symbol (?). For example, the filter, "Tag?" would search for all four character tagnames that begin with "Tag". The filter, "Tag??", would search for all five character tagnames that begin with "Tag", and so on.

Any sequence of valid OpenHMI tagname characters, together with the two wildcard characters, is acceptable in a filter. The valid tagname characters are: A-Z, a-z, 0-9, !, @, -, ?, #, \$, %, \_, \ and &.

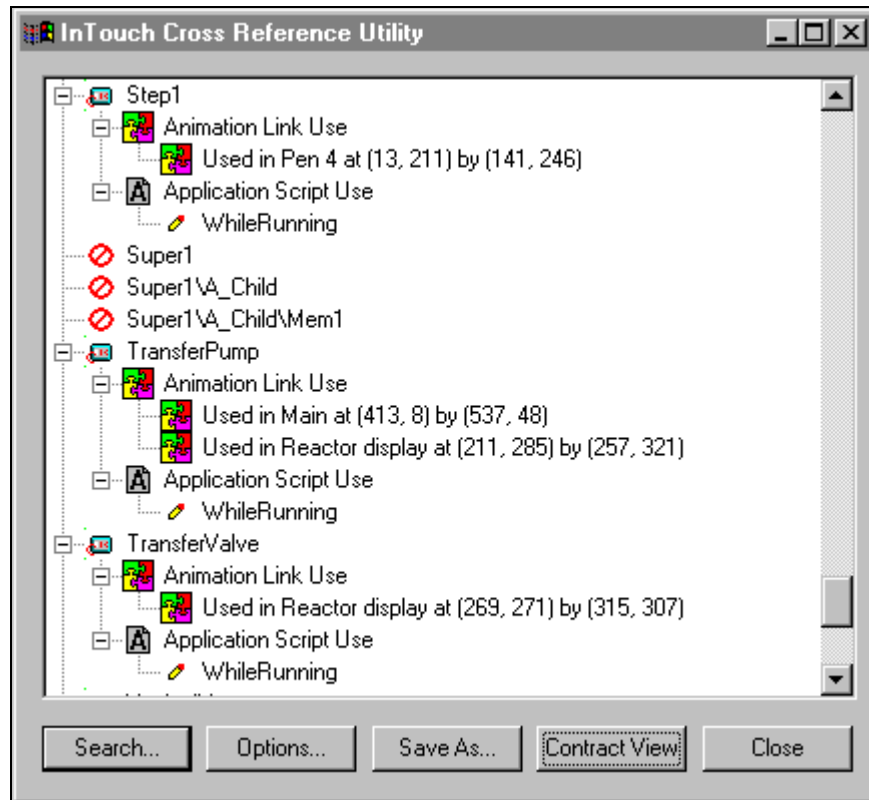
If you right click the mouse in the **Filter** box, a menu will appear displaying the commands that you can apply to the selected text.

4. Click **Search** to begin the cross reference search based upon your specified view criteria.

## Viewing the Cross Reference Search Results



























When you perform a cross reference search, the **OpenHMI Cross Reference Utility** dialog box appears listing all instances of usage found for the **Filter** that you specified.

If no filter is used, all tagnames defined in the current application's Tagname Dictionary are displayed. For example:



## Cross Reference Utility Icons

The following briefly describes the various icons that may appear in the OpenHMI Cross Reference Utility:

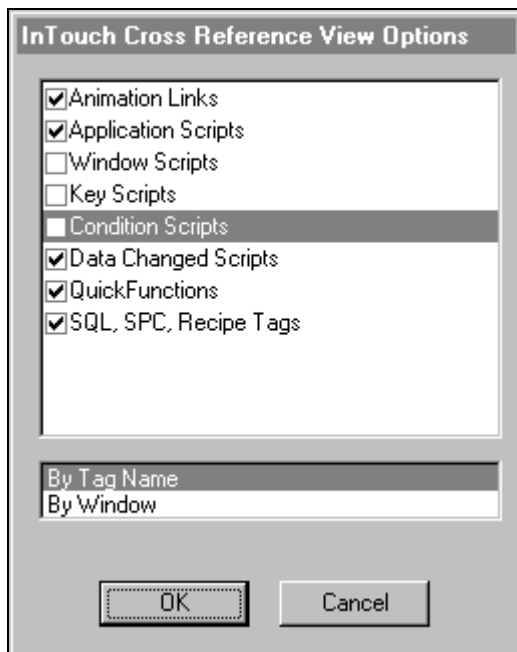
Icon	Description
	Tagname is assigned to an OpenHMI object, or used to store a value in an OpenHMI QuickScript, wizard or add-on program. Click to expand the level's view.
	Click to collapse an expanded level's view.
	Displayed tagname is defined in the application's Tagname Dictionary, but it is not assigned to an object.
	Displayed tagname is used in either an animation link or OpenHMI QuickScript. Double-click, or click  to expand the view.
	Displayed tagname is assigned to an animation link. Double-click, or click  to display the window name and the coordinates for object(s) in the window assigned to the animation link.
	Displayed tagname is used in an Application script. Double-click, or click  to expand the view and display the type of Application script that uses the tagname .
	Displayed for all Application <b>On Startup, While Running, and On Shutdown</b> scripts; Window <b>On Show, While Showing, and On Hide</b> scripts, and Key <b>On Key Down, While Down, and On Key Up</b> scripts. Double-click the script to view it.
	Displayed tagname is used in a Window script. Double-click, or click  to expand the view to display the name of the window with the script. Double-click any listed window name to view the script.
	Displayed tagname is used in a Data Change script. Double-click, or click  to expand the view, and then double-click any listed script to view it.
	Displayed tagname is used in a Condition script. Double-click, or click  to expand the view to display the script's condition and its type. For example, <b>\$Hour==12 On True</b> . Double-click any listed script to view it.
	Displayed tagname is used in a Key script. Double-click, or click  to expand the view and display the key assigned to the script and the script's type. For example, <b>F2 On Key Down</b> . Double-click any listed script to view it.
Ax	Displayed tagname is used in an ActiveX Event script. Double-click, or click  to expand the view and display the ActiveX Event script
	When cross referencing by <b>Window</b> , this icon precedes the window name in which the displayed tagname is used. Double-click, or click  to view all tagnames used in the window.
	Displayed tagname is used in a SPC Pro application. Double-click, or click  to view the name of the SPC Dataset in which the tagname is used.
	Displayed tagname is used in a SQL application. Double-click, or click  to view the name of the SQL Bind List in which the tagname is used.
	Displayed tagname is used in a Recipe Manager application.

## Changing the Cross Reference Search Criteria

If desired, after you have performed your initial cross reference search, you can narrow your search by modifying your original search options.

➤ **To change the search options:**

1. In the **OpenHMI Cross Reference Utility** dialog box, (displayed after you have performed your initial search), click **Options**. The **OpenHMI Cross Reference View Options** dialog box appears:

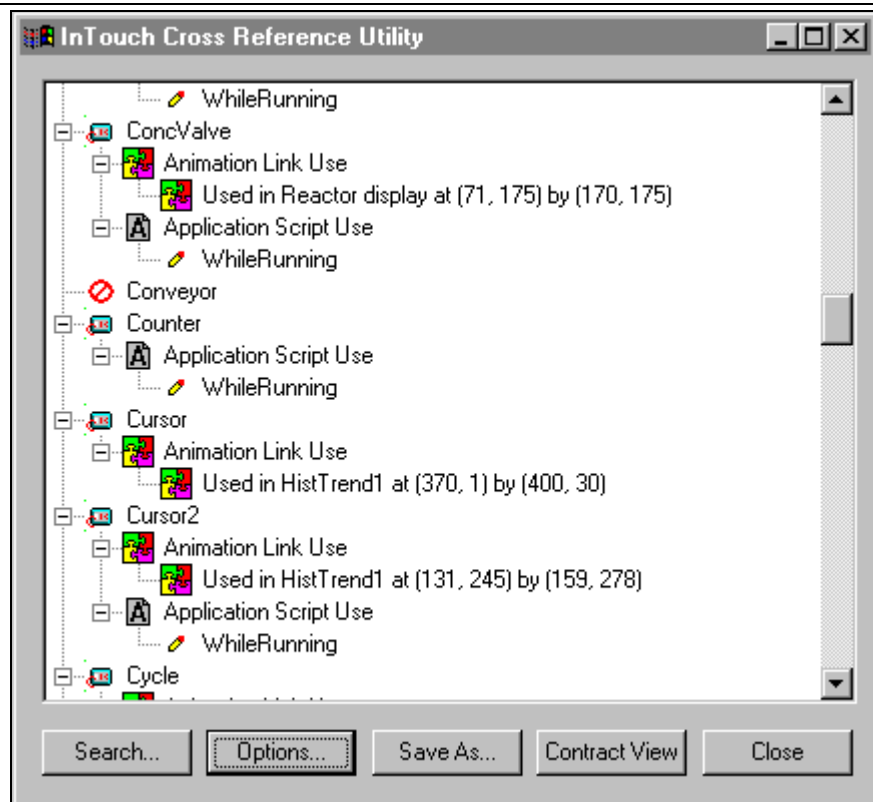


2. Select the search criteria options that you want to modify for your new search.
  - ☞ The options available here are based upon the **Search Criteria** you originally selected in the **OpenHMI Cross Reference Search Criteria** dialog box. If you selected **Search for all occurrences**, all search criteria options will be available. If you selected **Search for specific occurrences**, only the specific occurrences you originally selected will be available. To change your **Search Criteria** selection, click **Cancel**. The **OpenHMI Cross Reference Utility** dialog box will reappear. Click **Search**, and select the new **Search Criteria** option.
3. In the list at the bottom of the dialog box, select whether you want the tree view populated by tagname or window name, and then click **OK**.

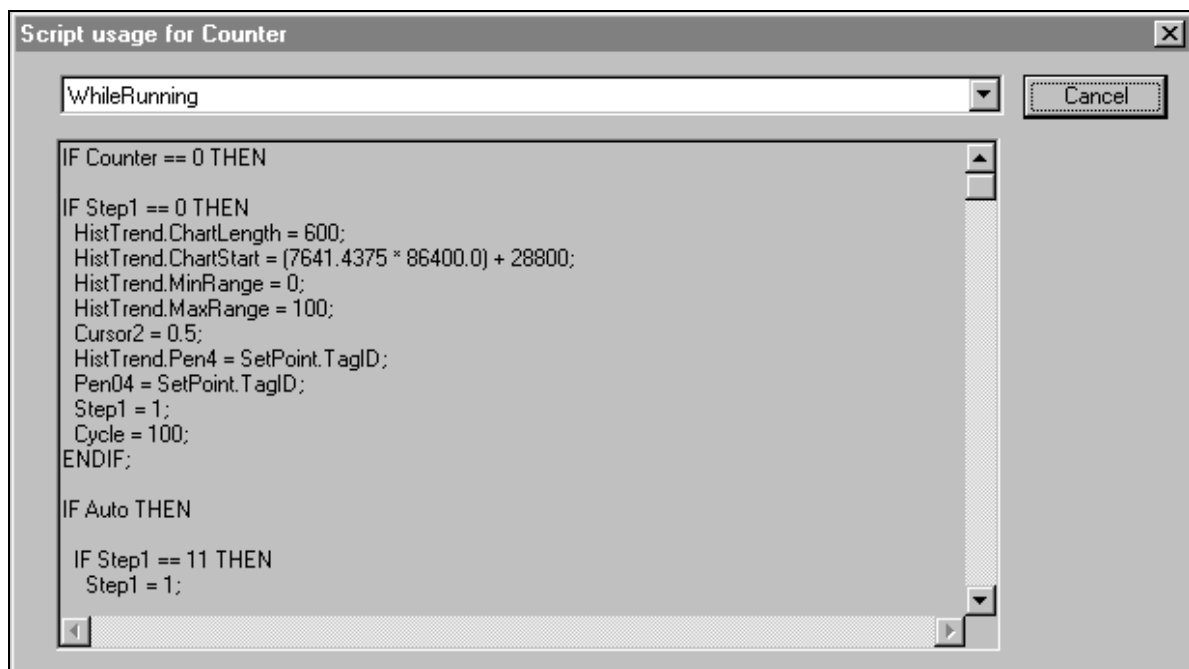
### Cross Referencing By Tagname

Alphabetically lists all tagnames found for you specified search criteria (default view). Based upon your specified search criteria, this view allows you to view the usage of all tagname found in windows, animation links, scripts and add-on applications.

- ☞ You can double-click a displayed tagname, and then double-click **Animation Link Use** to expand the view. When you expand the view, the window name and the location (coordinates) of the object(s) linked to the tagname are displayed. For example:



- You can double-click a tagname, and then double-click any of its associated scripts to open it in the **Script usage for Tagname** dialog box:



The list box at the top of the screen shows all scripts associated with the selected tagname. Click the arrow to open the list to select another script for viewing. For Application, Window, Key and Condition scripts, the list will contain the names of all scripts that use this tagname. For Data Change scripts, only the tagname is listed.

Click **Cancel** to close the dialog box dialog box.



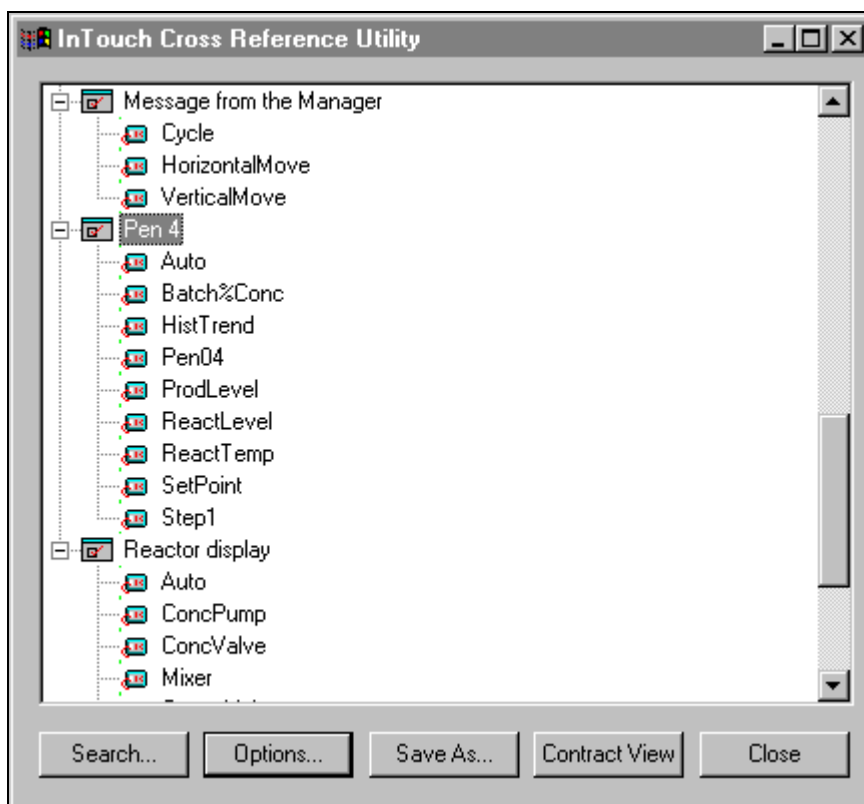
**Note** This is a read only view of the QuickScript. You cannot edit the QuickScript text in this dialog box. However, you can copy any portion or all of the QuickScript, and then paste it into any OpenHMI QuickScript editor window.

To copy the QuickScript to the Windows Clipboard, right-click the script, then click **Select All**. Right-click the script again and click **Copy**. You can also execute the Windows copy command (CTRL+C).

To paste the copied script into another OpenHMI QuickScript, in the Application Explorer under **Scripts**, double-click the type of script that you want to create. The QuickScript editor will appear. On the **Edit** menu, click **Paste**, or right-click the script window, and then click **Paste**. You can also execute the Windows paste command (CTRL+V).

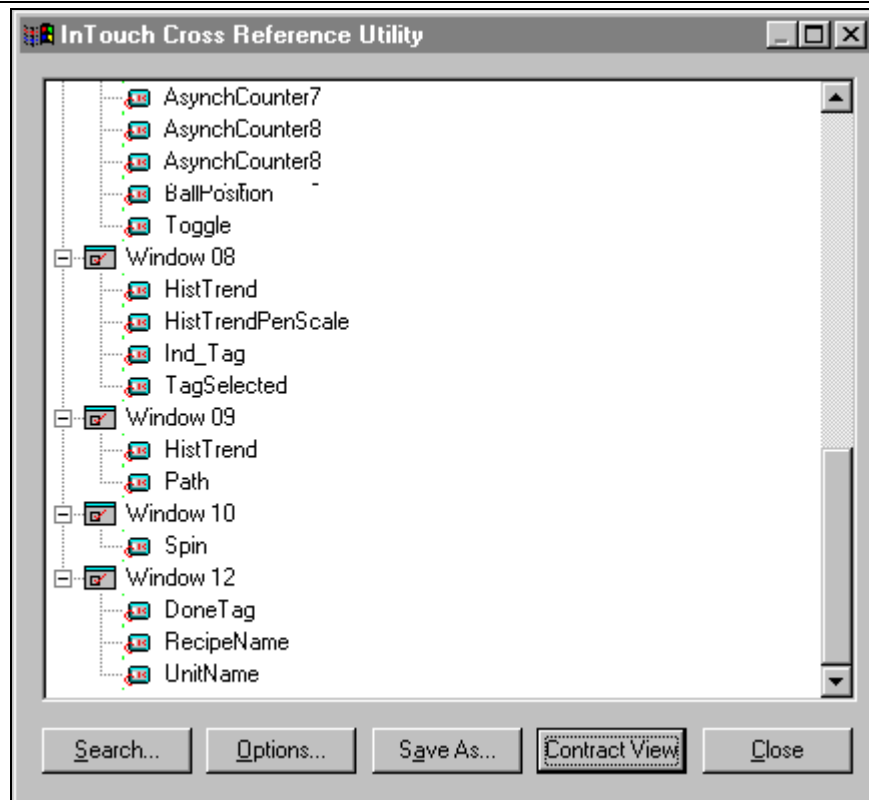
## Cross Referencing By Window Name

Sorts the display by window name then the tagnames used in the window. For example:



**Note** This view only displays the tagnames used in the window, it does not include usage in animation links, scripts, and so on.

Click **Expand View** to display all view levels available for the displayed tagnames or windows. For example:



Click **Contract View** to return the dialog box to its default mode.

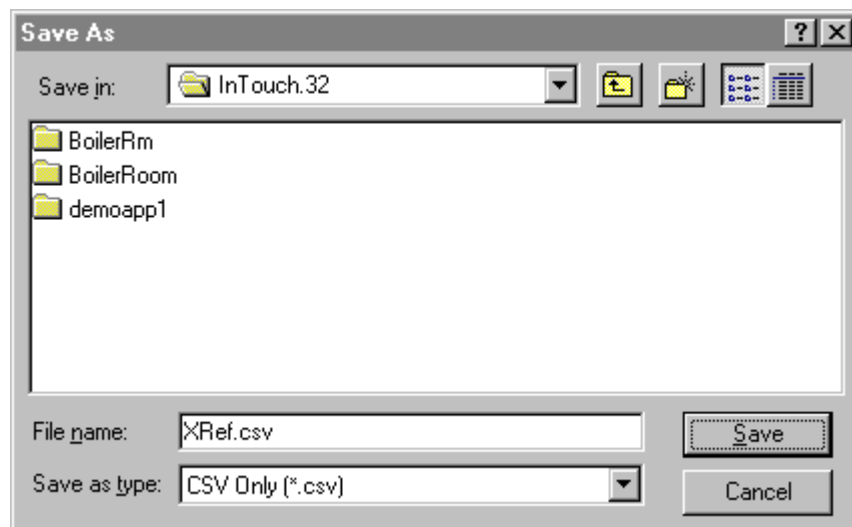
Click **Close** to exit the cross reference utility.

## Saving Cross Reference Files

Your cross reference files can be saved and view later in any text editor program that supports the comma separated variable (.CSV) file. The information stored in a cross reference file corresponds to the information currently displayed in the **OpenHMI Cross Reference Utility** dialog box.

➤ **To save a cross reference file:**

1. In the **OpenHMI Cross Reference Utility** dialog box, click **Save As**. The **Save As** dialog box appears:

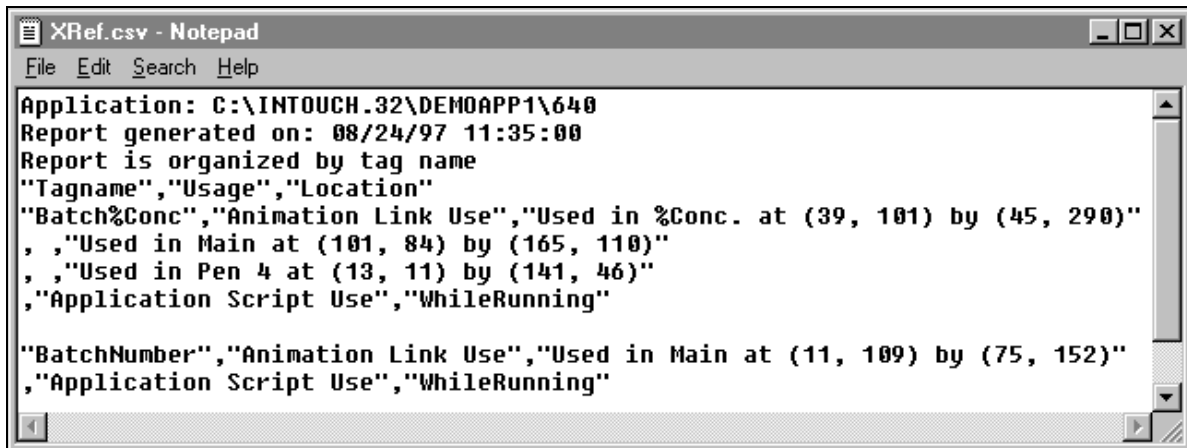


- In the **File name** box, type the name that you want to save the cross reference file under.
  - The file must be save as a .CSV file.
- Click **Save**.

## Printing Cross Reference Files

You can open a cross reference .CSV file in any text editor program that supports the .CSV file format and print the cross reference file as a report.

For example, if you open the file in Notepad, it would appear as follows:

A screenshot of a Notepad window titled "XRef.csv - Notepad". The window has a menu bar with "File", "Edit", "Search", and "Help". The text content is as follows:

```
Application: C:\INTOUCH.32\DEMOAPP1\640
Report generated on: 08/24/97 11:35:00
Report is organized by tag name
"Tagname","Usage","Location"
"Batch%Conc","Animation Link Use","Used in %Conc. at (39, 101) by (45, 290)"
, ,"Used in Main at (101, 84) by (165, 110)"
, ,"Used in Pen 4 at (13, 11) by (141, 46)"
,"Application Script Use","WhileRunning"

"BatchNumber","Animation Link Use","Used in Main at (11, 109) by (75, 152)"
,"Application Script Use","WhileRunning"
```

To print the file, on the **File** menu, click **Print**.

## Printing Tagname Dictionary Details

In addition to printing a saved cross reference .CSV file, you can print listings of the Tagname Dictionary details, alarm information, link details and scripts. Printing the Tagname Dictionary details can help you to determine the usage of tagnames.

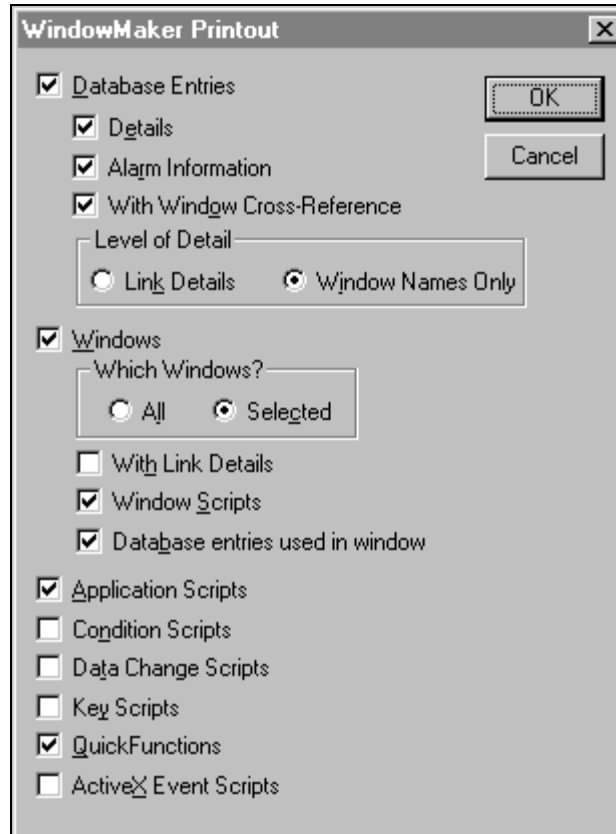
---

**Note** Your Windows default printer will be used to produce the printout which will be 80 columns wide. Your default printer is selected and setup through the Windows Control Panel.

---

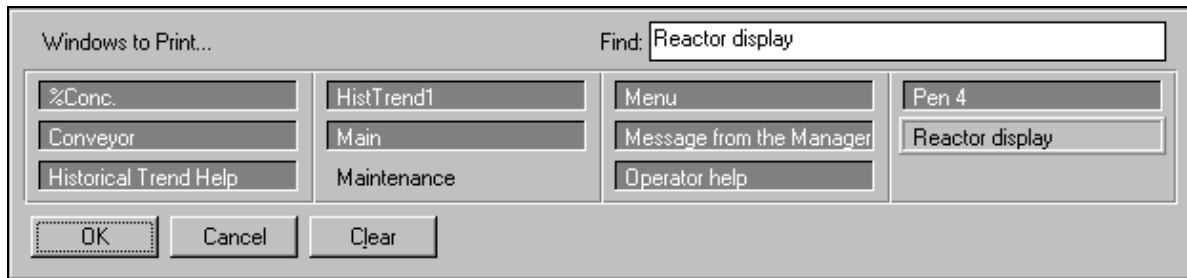
➤ **To print Tagname Dictionary details:**

1. On the **File** menu, click **Print**. The **WindowMaker Printout** dialog box appears:



2. Select **Database Entries** if you want to print all database information. If you select **Database Entries**, the following three options become active:
  - Select **Details** to include the database details in your report.
  - Select **Alarm Information** to include the database alarm information in your report.
  - Select **With Window Cross Reference** to print all database entries with window cross-references. Selecting this option will activate **Level of Detail** options:
    - Select **Link Details** to print the location and animation link details where the tagname was used.
    - Select **Window Names Only** to print only the name of the cross-referenced windows(s).
3. Select **Windows** to print a listing of the database entries used in the application windows. If you select **Windows**, the following three options become active:

- Select **All** to print the database entries for all windows in the application.
- Select **Selected** to print only the database entries for specific windows. The **Windows to Print** dialog box appears:



4. Select the windows you want to print, then click **OK**. (By default, all window names will be selected when the dialog box appears.)
  - Select **With Link Details** to print the link details for the window(s).
  - Select **Window Scripts** to print the scripts associated with the window(s).
  - Select **Database entries used in window** to print the tagnames used in the window(s).
  - Select **Application Scripts** to print the application scripts.
  - Select **Condition Scripts** to print the condition scripts associated with the window(s).
  - Select **Data Change Scripts** to print the data change scripts associated with the window(s).
  - Select **Key Scripts** to print the key scripts associated with the window(s).
5. Click **OK** to begin printing your report.

## Deleting Tagnames from the Dictionary

OpenHMI maintains a use count for each item in the database. This count is not updated automatically for certain operations such as, deleting a window, changing tagnames in links or scripts, and so on. In these cases, OpenHMI continues to consider the tagname as being used in the application and will not allow you to delete it. Therefore, you may need to update your use count in order to delete a tagname.

➤ **To delete an unused tagname:**

1. Close WindowViewer if it is running.
2. On the **Special** menu, click **Tagname Dictionary**. The **Tagname Dictionary** dialog box will appear.
3. Click **Select**. The **Select Tagname** (Tag Browser) will appear.
4. Select the tagname that you want to delete then click **OK**. The **Tagname Dictionary** dialog box will appear displaying the selected tagname's definition.
5. Click **Delete**.

---

**Note** The **Delete** button will not be available if WindowViewer is running or OpenHMI considers the tagname as being used in the application.

You can determine where a tagname is being used through the OpenHMI cross reference utility. (On the **Special** menu, click **Cross Reference**.) Or, you can print a report of all tagname links used in a window. (On the **File** menu, click **Print**.)

---

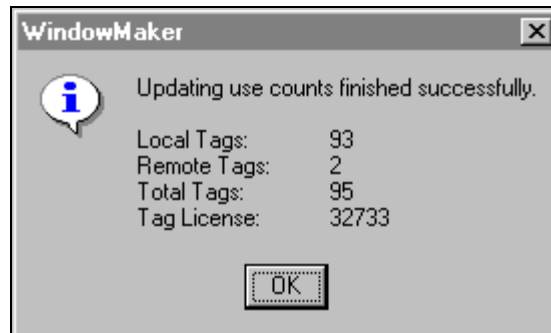
☞ For more information on printing reports, see "Printing Tagname Dictionary Details."

## Updating Use Counts

Since OpenHMI maintains a use count for each item in the database you may need to update the use counts to set all unused tagnames to zero before OpenHMI will allow you to delete any of them.

➤ **To update tagname use counts:**

1. Close all your windows.
2. On the **Special** menu, click **Update Use Counts**.
  - ☞ A message box will appear telling you that updating use counts can take quite a while. You may at that time cancel the command or continue.
3. Click **Yes** to continue updating the use counts. Once the system has completed updating the use counts, the following dialog box appears:



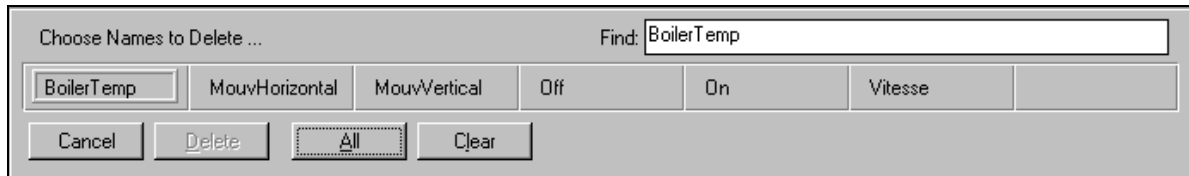
4. Click **OK**.

## Deleting Unused Tagnames

After you have updated the use count, OpenHMI will allow you to delete all unused tagnames. You can either delete them by opening each of them in the **Tagname Dictionary** dialog and clicking **Delete** or, you can delete one or more of them at once by using the **Delete Unused Tags** command.

➤ **To delete multiple unused tagnames:**

1. On the **Special** menu, click **Delete Unused Tags**. The **Choose Names to Delete** dialog box appears:



2. Select the tagnames that you want to delete, then click **Delete**.
3. Click **All** to delete all tagnames displayed.

---

**Warning!** Tagnames that are only alarmed have no use count, and can be accidentally deleted. To ensure that alarmed only tagnames are included in the use count, you need to use them in a window or QuickScript.

---

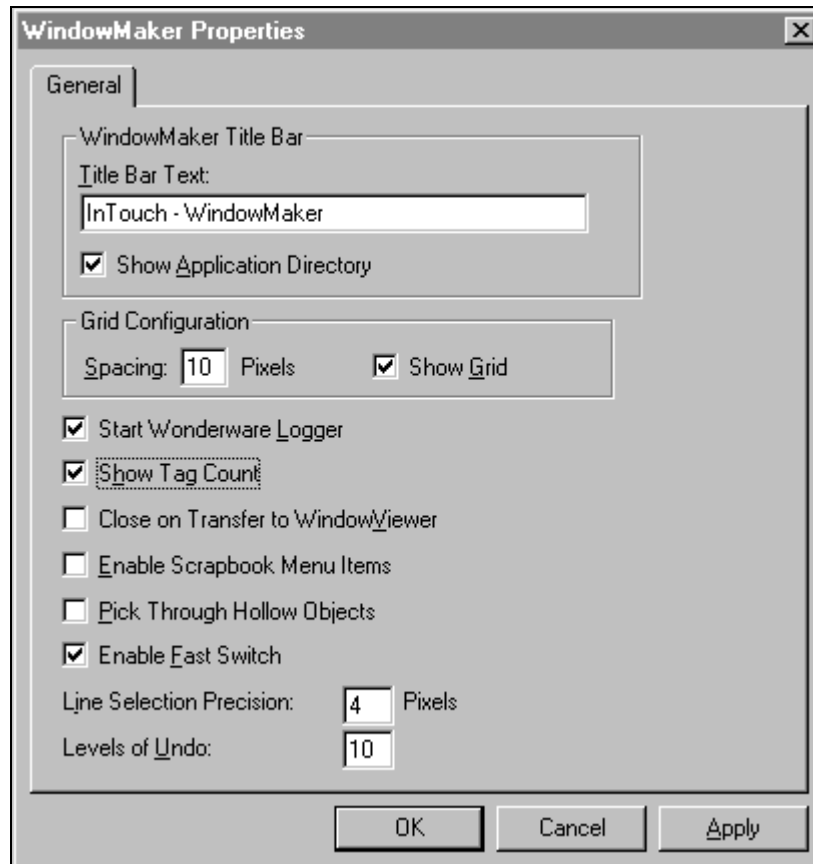
## Displaying the Tag Usage Count

You can display the number of local tagnames that are currently defined in your Tagname Dictionary in the menu bar in WindowMaker. (The tagname count does not include internal system tagnames.)

➤ **To show the tagname count:**

1. On the **Special** menu, point to **Configure**, and then click **WindowMaker**. The **WindowMaker Properties - General** property sheet appears:

☞ To quickly access the dialog box, in the Application Explorer under **Configure**, double-click **WindowMaker**.



2. Select **Show Tagname Count**.
3. Click **OK**.
4. The total number of local tagnames defined in your Tagname Dictionary will now be displayed at the end of your WindowMaker menu bar.

☞ The entire Tagname Dictionary must be read in order to update the displayed tagname count. Therefore, when this option is turned on, performance may be degraded when you are making changes to your Tagname Dictionary. If your Tagname Dictionary is large, you should not select this option.



## Substituting Tagnames

When you duplicate an object it is an exact replica of the original including links, animation, scripts and so on. However if you need to use a different tagname on an object that you have duplicated you must change the tagname. In WindowMaker, this is called "substituting a tagname." You can select and change the tagnames for any object at any time and you can select multiple objects and change all their tagnames the same time.

☞ If you change a tagname for an object and WindowViewer is running, you will need to restart WindowViewer for the change to take effect.

➤ **To change an object's tagname to another local tagname:**

1. Select the object(s) whose tagname you want to change, and then on the **Special** menu, click **Substitute Tags**. The **Substitute Tagnames** dialog box appears:

☞ To quickly access the dialog box, right-click one of the selected objects, point to **Substitute**, and then click **Substitute Tags**.

Current Name:	Required Type	New Name:
Compressor	Discrete	Compressor
WaterHeater	Discrete	WaterHeater
WaterPump	Discrete	WaterPump

Buttons: OK, Cancel, Index, Convert, Replace

2. In the **New Name** box, enter a new tagname, and then click **OK**. The tagname associated with the selected object(s) will automatically be changed.

☞ If right-click the **New Name** box, a menu will appear displaying the commands that you can apply to your text.

If you double-click a tagname in the **New Name** box, its definition in the Tagname Dictionary will appear.

If you erase the tagname then double-click in the blank **New Name** box, the Tag Browser will appear.

## Converting Placeholder Tagnames

When you index tagnames (to take them out of service) or you import or export a window or QuickScript to or from your current application, all the tagnames associated with that window or QuickScript are transferred with the window, but they are not added to your new application's database. Instead, they are automatically marked as "placeholder" (index) tagnames. You must convert these placeholder tagnames and, if required, define them in your new application Tagname Dictionary. For example:

The screenshot shows a dialog box titled "Substitute Tagnames..." with "1 of 4" in the top right corner. The dialog contains a table with three columns: "Current Name:", "Required Type", and "New Name:". Below the table are five buttons: "OK", "Cancel", "Index", "Convert", and "Replace". The "Convert" button is highlighted with a dashed border.

Current Name:	Required Type	New Name:
?d:WaterHeater	Discrete	?d:WaterHeater
?i:WaterPump	Analog	?i:WaterPump
?m:WarningMsg	String	?m:WarningMsg
?r:Compressor	Analog	?r:Compressor

In this example, to convert the placeholder tagnames to local tagnames, click **Convert**.

- When you import a window, if any of the tagnames are not defined in your local Tagname Dictionary, you will be prompted to define each of them before you can convert them. If this is the case, click **OK**. The **Tagname Dictionary** dialog box will appear and you can define the tagname(s).

Notice the placeholders **?d:**, **?i:**, **?m:** and **?r:** preceding the tagnames. They indicate the type that the tagname was originally defined as:

- d** Discrete type
- i** Integer type
- m** Message type
- r** Real type

## Scaling I/O Tagnames

All I/O type tagnames receive their values from other Windows application programs such as Excel and I/O Servers. This value is referred to as the "raw" value. When you define a tagname in the Tagname Dictionary, you must enter values for the "Min Raw" and "Max Raw." These values are used by the database as clamps on the actual raw value received from the I/O device. For example, if you set the "Min Raw" value to 50 and the actual value received from a I/O Server is 0, database will force the Raw value to 50.

OpenHMI does not display raw values. Instead, it displays engineering units (EU). When you define an I/O type tagname in the Tagname Dictionary, you must specify values for the "Min EU" and "Max EU." These values are used to scale the raw value to the displayed value. If you do not want to do scaling or your I/O device does the scaling for you, set the Min/Max EU values equal to the Min/Max Raw values.

For example, let's assume that a flow transmitter wired to a PLC register generates a value of zero at no flow and a value of 9999 at 100% flow. The following values would be entered:

```
Min EU = 0           Max EU = 100
Min Raw=0           Max Raw = 9999
```

A raw value of 5000 would be displayed as 50.

Let's also assume that a flow transmitter wired to a PLC register generates a value of 6400 at no flow and a value of 32000 at 300 GPM.

```
Min EU = 0           Max EU = 300
Min Raw = 6400       Max Raw = 32000
```

In this case, a raw value of 12800 would be displayed as 150. A raw value of 6400 would be displayed as 0 and a Raw value of 0 would be displayed as 0 (all values outside the boundaries set by the Min Raw and Max Raw values are clamped).

The above scaling works in reverse when the I/O tagname data is written from the OpenHMI Tagname Dictionary to other Windows applications.

For example, an operator could enter a setpoint value of 0-300 GPM in a data entry window and have a value of 6400-32000 transmitted to the PLC register.

This is always valid when the OpenHMI program is acting as the client (either requesting data from or sending data to another Windows program). However, when OpenHMI acts as data source (another Windows program is requesting data from OpenHMI) it will return the EU value to the requesting program.

For example, if a cell in an Excel spreadsheet contained the remote reference formula:

```
=view|tagname!speed
```

The value displayed in the cell would be the current EU value for the tagname speed.

## Instrument Failure Monitoring

Beginning with Version 7.0, OpenHMI supports three tagname **.fields** (**.RawValue**, **.MinRaw** and **.MaxRaw**) that you can use in OpenHMI QuickScripts to monitor instrument values to determine out-of-range, out-of-calibration or, failure.

# Internal System \$Tagnames

OpenHMI provides you with numerous predefined internal system tagnames that you can use to perform a variety of actions. For example, if you want to display the current time, you could link the system tagname **\$TimeString** to a value display string. All internal tagnames are preceded with a dollar sign (\$). The internal system tagnames are accessed through the Tag Browser.


*☞* For more information, see "The Tag Browser."

The following section briefly describes the internal system tagnames:

System Tagname	Description
<b>\$AccessLevel</b>	Read only integer security tagname used in expressions or scripts to control the operator's ability to perform specific functions.
<b>\$AlarmPrinterError</b>	Read only discrete tagname that is equal to 1 if there is a printer error.
<b>\$AlarmPrinterNoPaper</b>	Read only discrete tagname that is equal to 1 if the printer is out of paper.
<b>\$AlarmPrinterOffline</b>	Read only discrete tagname that is equal to 1 if the printer is offline.
<b>\$AlarmPrinterOverflow</b>	Read only discrete tagname that is equal to 1 if there is printer overflow.
<b>\$ApplicationChanged</b>	Read only real tagname that reflects whether or not the remote application has changed in distributed systems. This number is incremented each time the <b>Notify Clients</b> command is selected on the Server node's WindowViewer <b>Special</b> menu.
<b>\$ApplicationVersion</b>	Read only real tagname that reflects the current version number of the application. This number changes each time a tagname or QuickScript is changed, added or deleted.
<b>\$ChangePassword</b>	Write only discrete security tagname that allows the operator to set the value of the <b>\$ChangePassword</b> tagname to 1, causing the generic Change Password dialog box to be displayed for the operator.
<b>\$ConfigureUsers</b>	Write only discrete security tagname that can be used on a discrete button to allow the operator to set the value of the <b>\$ConfigureUsers</b> tagname to 1, causing the generic <b>Configure Users</b> dialog box to be displayed for editing the security user name list.

System Tagname	Description
<b>\$Date</b>	Read only integer tagname that displays the whole number of days which have passed since 1/1/70.
<b>\$DateString</b>	Read only memory message tagname that displays the date in the same format set in the WIN.INI file, for example, 4/18/1992. (This date format is set through the Windows Control Panel.)
<b>\$DateTime</b>	Read only real tagname that displays the fractional number of days which have passed since 1/1/70.
<b>\$Day</b>	Read only integer tagname that displays the current day (value may be 1-31).
<b>\$Hour</b>	Read only integer tagname that displays the current hour of the day (value may be 0-23).
<b>\$InactivityTimeout</b>	Read only discrete security tagname that equals 1 when the time configured for automatic log off of the operator has elapsed.
<b>\$InactivityWarning</b>	Read only discrete security tagname that equals 1 when the time configured for warning the operator that log off is about to occur has elapsed.
<b>\$LogicRunning</b>	Read/write discrete tagname used to monitor and/or control the running of scripts.
	<b>Note</b> Asynchronous User Defined Function scripts that are currently executing cannot be stopped. However, you can prevent new scripts from executing.
<b>\$Minute</b>	Read only integer tagname that displays the current minute (value may be 0-59).
<b>\$Month</b>	Read only integer tagname that displays the current month (value may be 1-12).
<b>\$Msec</b>	Read only integer tagname that displays milliseconds (value may be 0-999).
<b>\$NewAlarm</b>	Read/write discrete tagname that is equal to 1 each time a new alarm occurs.
<b>\$ObjHor</b>	Read only integer tagname used to display the horizontal pixel location of the center of a selected object.
<b>\$ObjVer</b>	Read only integer tagname used to display the vertical pixel location of the center of a selected object.

System Tagname	Description
<b>\$Operator</b>	Read only security message tagname that can be used in an expression or QuickScript to control the operator's ability to perform specific functions.
<b>\$OperatorEntered</b>	Read/write security message tagname that sets the "User Name" for the operator.
<b>\$PasswordEntered</b>	Write only security message tagname that sets the "Password" for the operator.
<b>\$Second</b>	Read only integer tagname that displays the current seconds (value may be 0-59).
<b>\$StartDdeConversations</b>	Read/write discrete tagname used to start uninitiated conversations during runtime when the <b>Special</b> menu has been disabled.
<b>\$System</b>	Read only Alarm Group type tagname for the alarm root group. If a tagname is not assigned to a specific Alarm Group name, it is automatically assigned to this root group by default. All defined Alarm Groups are descendants of <b>\$System</b> .
<b>\$Time</b>	Read only integer tagname that displays the time in milliseconds since midnight.
<b>\$TimeString</b>	Read only memory message tagname that displays the time in the same format set in the WIN.INI file, e.g., 12:01:59 PM. (This time format is set through the Windows Control Panel.)
<b>\$Year</b>	Read only integer tagname that displays the year in four digits, e.g., 1990.

 For more information on system tagnames, see your *OpenHMI Reference Guide*.

# Tagname .Fields

Most discussions concerning OpenHMI refer to the concept of objects. The concept of objects is very widespread and complex. For our discussion we will limit our definition of an object to a collection of information about a graphical object on the screen or information about a tagname in the Tagname Dictionary.

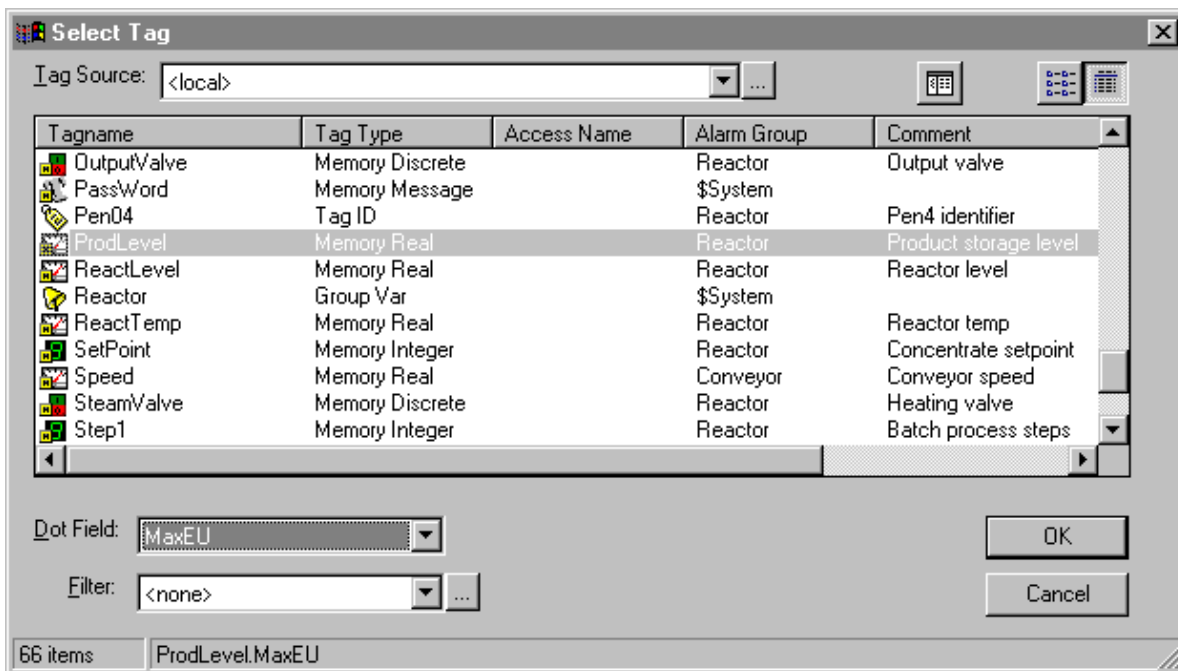
For example, if a rectangle is drawn on the screen it has certain "attributes" such as the line width, line and fill color, pixel location on the screen, links associated with, etc. Tagnames work in much the same way. For example, if an analog, alarmed tagname is created called "Analog\_Tagname," it will have "attributes" associated with it such as the name of the tagname, the tagname's **HiHi** alarm setpoint, and so on. Some of these "attributes" are accessible within OpenHMI through scripts, expressions and user inputs and are known as **.fields**. The syntax required to access these data fields associated with a tagname is **Tagname.field**.

For example, to allow runtime changes to the **HiHi** alarm limit on tagname "Analog\_Tagname," you could create an **Analog - User Input** touch link and apply it to a button that is defined with the expression **Analog\_Tagname.HiHiLimit**. In runtime, the operator would simply click the button and type in a new value for the **HiHi** alarm limit being used for "Analog\_Tagname."

You can use **.fields** to allow input and output of data associated with a tagname.

➤ **To access tagname .fields:**

1. Type a tagname plus a period (**tagname.**) in any OpenHMI QuickScript, or animation link tagname or expression input box, and then double-click to the right of the period (.). The Tag Browser appears displaying the tagnames defined for the current tag source:



➤ **To select a .field:**

1. Click the **Dot Field** arrow to open the list of **.fields** that you can associate with the type of tagname currently selected.
  - ☞ By default, **<none>** will initially be displayed for all types of tagnames.

**Note** **Dot Field** is not available when you access the Tag Browser from the Tagname Dictionary.

- Click the **.field** in the list that you want to append to the selected tagname.

**Note** Not every tagname type has the same **.fields**. For example, a **Discrete** type tagname has **.OnMessage**, whereas an analog does not. If you select a **Discrete** type tagname and you assign **.OnMessage** to it, and then you select another **Discrete** type tagname, the displayed **.field** list will not change. But, if you select another type of tagname in the control view list, for example an analog, the displayed **.field** will revert to **<none>**.

*ℹ* For more information on the Tag Browser, see "The Tag Browser."

The following section briefly describes the tagname **.fields**:

<b>.Field</b>	<b>Description</b>
<b>.Ack</b>	Read/write discrete tagname .field that monitors/controls the alarm acknowledgment status of tagnames, Alarm Groups and/or Group Variables.  <b>Note</b> <b>.Ack</b> has an inverse tagname .field called <b>.Unack</b> . When an unacknowledged alarm occurs, <b>.Unack</b> will be set to 1. <b>.Unack</b> can then be used in animation links or condition scripts to trigger annunciators for any unacknowledged alarms.
<b>.Alarm</b>	Read only discrete tagname .field that is equal to 1 when an alarm condition exists for the specified tagname, Alarm Group Name or Group Variable.
<b>.AlarmDevDeadband</b>	Read/write analog (only valid for integer or real tags) tagname .field that monitors/controls the deviation percentage deadband for both minor and major deviation alarms.
<b>.AlarmEnabled</b>	Read/write discrete tagname .field that disables/enables events and alarms for a tagname, Alarm Group or Group Variable.
<b>.AlarmValDeadband</b>	Read/write analog (only valid for integer or real tags) tagname .field that monitors and/or controls the value of an alarm's deadband. This field is valid for Alarm Groups and Group Variables as well as ordinary tags.



---

<b>.Field</b>	<b>Description</b>
<b>.Comment</b>	Read only message tagname tagname .field that is used to display the comment field entered for a tagname in the Tagname Dictionary.
<b>.DevTarget</b>	Read/write analog (only valid for integer or real tags) tagname .field that monitors and/or controls the target for minor and major deviation alarms.
<b>.DisplayMode</b>	Read/write analog tagname .field used to determine the method to be used in displaying values on the trend.
<b>.EngUnits</b>	Read/write analog tagname .field used to access the engineering units of an analog tagname as specified in the tagname dictionary. Note, writing to these values is not retentive.
<b>.HiLimit, .HiHiLimit, .LoLimit, .LoLoLimit</b>	Read/write analog tagname .field that monitors/controls the limits for value alarm checks. These .fields are only valid for integer and real tags.
<b>.HiStatus, .HiHiStatus, .LoStatus, .LoLoStatus</b>	Read only discrete tagname .field that determines whether an alarm of a specified type exists.
<b>.MajorDevPct</b>	Read/write real tagname .field that monitors or controls the major percentage of deviation for alarm checking.
<b>.MajorDevStatus</b>	Read only discrete tagname .field that determines whether a major deviation alarm exists for the specified tagname.
<b>.MaxEU, .MinEU</b>	Read only interger tagname .field that display the maximum and minimum values for the tagname.

<b>.Field</b>	<b>Description</b>
<b>.MaxRange, .MinRange</b>	Read/write real tagname .field used to represent the percentage of the tagname's Engineering Unit range that should be displayed for each tagname being trended. The limits for <b>.MaxRange</b> and <b>.MinRange</b> are from <b>0</b> to <b>100</b> and <b>.MinRange</b> should always be less than <b>.MaxRange</b> . If a value less than <b>0</b> or greater than <b>100</b> is assigned to either of these fields, the value will be clamped at <b>0</b> or <b>100</b> . If <b>.MinRange</b> is greater than or equal to <b>.MaxRange</b> , the trend will not display any data.
<b>.MaxRaw, .MinRaw</b>	The high/low clamp setting for the actual raw value received from an I/O Server by WindowViewer as a client. The value for <b>.MaxRaw/.MinRaw</b> tagname .field comes from the Max/Min Raw value field in tagname database for the specified I/O tagname. Any raw value which exceeds or falls below this setting is clamped to this value.
<b>.MinorDevPct</b>	Read/write real tagname .field used to monitor and/or controls the minor percent of deviation for alarm checking.
<b>.MinorDevStatus</b>	Read only discrete tagname .field used to determine whether a minor deviation alarm exists for the specified tagname.
<b>.Name</b>	Read message tagname .field used to display the actual name of the tagname. For example, it can be used to determine the name of an Alarm Group that a Group Variable is pointing to, or the name of a TagID tagname. It can also be written to in order to change the Alarm Group that a Group Variable is pointing to.
<b>.Normal</b>	Read only discrete tagname .field that is equal to 1 when there are no alarms for the specified name. This .field is valid for Alarm Groups and Group Variables as well as ordinary tagnames.
<b>.OffMsg, .OnMsg</b>	Read/write message tagname .field used to display the on message and off message specified in the Tagname Dictionary for a discrete tagname.

---

**Note** writing to these values is not retentive.

---

.Field	Description
<b>.Quality</b>	Read only interger tagname allows the user to access the quality of an I/O tagname as provided by an I/O server.
<b>.QualityLimit</b>	Read only integer tagname .field used to display the quality limit of an I/O value provided by data source when the I/O connection is valid.
<b>.QualityLimitString</b>	Read only message tagname .field used to display the quality limit string of an I/O value provided by an I/O server when the I/O connection is valid.
<b>.QualityStatus</b>	Read only integer tagname .field used to display the quality status of an I/O value provided by an I/O server when the I/O connection is valid.
<b>.QualityStatusString</b>	Read only message tagname .field used to display the quality status string of an I/O value provided by an I/O server when the I/O connection is valid.
<b>.QualitySubstatus</b>	Read only integer tagname .field used to display the quality substatus of an I/O value provided by an I/O server when the I/O connection is valid.
<b>.QualitySubstatusString</b>	Read only message tagname .field used to display the quality substatus string of an I/O value provided by an I/O server when the I/O connection is valid.
<hr/> <p><b>Note</b> If the I/O connection becomes invalid, the quality .fields are automatically reset to the initial value of zero. The <b>.ReferenceComplete</b> flag is also set to zero at this time.</p> <hr/>	
<b>.RawValue</b>	Any type (Real/discrete) tagname .field that is used to display the actual discrete or analog I/O value before OpenHMI applies scaling.
<b>.Reference</b>	Read/write tagname .field used with I/O type tagnames to dynamically change the address of the tagname's source.
<b>.ReferenceComplete</b>	Discrete tagname .field that provides a confirmation that the item requested is the same reflected in the <b>.Value</b> field.
<p><i>↪</i> For more information on the .Reference .field, see "Dynamic Reference Addressing."</p>	
<b>.ROCPct</b>	Read/write tagname .field used to monitor and/or control the rate of change for alarm checking.
<b>.ROCStatus</b>	Read only discrete tagname .field used to determine whether a Rate-of-Change alarm exists for the specified tagname.

<b>.Field</b>	<b>Description</b>
<b>.TimeDate</b>	Real integer tagname .field used to display the whole number of days which have passed since an I/O value was provided by an I/O server when the I/O connection was valid.
<b>.TimeDateString</b>	Read only message tagname .field used to displays the date in the same format set in the WIN.INI file. For example, 10/31/1997 that an I/O value was provided by an I/O server when the I/O connection was valid.
<b>.TimeDateTime</b>	Read only real tagname .field used to display the fractional number of days which have passed since an I/O value was provided by an I/O server when the I/O connection was valid.
<b>.TimeDay</b>	Read only integer tagname .field used to display the day an I/O value was provided by an I/O server when the I/O connection is valid. (value may be 1-31).
<b>.TimeHour</b>	Read only integer tagname .field used to display the hour of the day that an I/O value was provided by an I/O server when the I/O connection was valid. (value may be 0-23).

<b>.Field</b>	<b>Description</b>
<b>.TimeMinute</b>	Read only integer tagname .field used to display the minute that an I/O value was provided by an I/O server when the I/O connection was valid. (value may be 0-59).
<b>.TimeMonth</b>	Read only integer tagname .field used to display the month that an I/O value was provided by an I/O server when the I/O connection was valid. (value may be 1-12).
<b>.TimeMsec</b>	Read only integer tagname .field used to display the time in milliseconds that an I/O value was provided by an I/O server when the I/O connection was valid.
<b>.TimeSecond</b>	Read only integer tagname .field used to display the time in seconds that an I/O value was provided by an I/O server when the I/O connection was valid. (value may be 0-59).
<b>.TimeTime</b>	Read only integer tagname .field used to display the time in milliseconds (since midnight) that an I/O value was provided by an I/O server when the I/O connection was valid.
<b>.TimeTimeString</b>	Read only message tagname .field used to display the time and day of an I/O value provided by an I/O server when the I/O connection is valid.
<b>.TimeYear</b>	Read only integer tagname .field used to display the time in the same format set in the WIN.INI file (for example, 12:09:45) that an I/O value was provided by an I/O server when the I/O connection was valid. (This time format is set through the Windows Control Panel.)
<b>.Unack</b>	Read/write discrete tagname .field used to control the alarm unacknowledgement status of local alarm(s).
<b>.UpdateCount</b>	Read-only integer tagname .field that is incremented when a retrieval is complete for the trend.
<b>.Value</b>	Read or Read/write analog tagname .field that displays the value of the specified tagname.

## Addressing Bit Fields for Analog Tagnames

Single bits within an integer tagname can be addressed by using the bit .field. These are all considered to be discrete (0/1) and if written, the analog tagname is promptly updated. You can use the bit .fields anywhere that a discrete tagname can be used. For example, in I/O, scripts, expressions, and so on.

- .00** Least significant bit
- .01** next more significant bit
- .02** etc.
- .**
- .**
- .**
- .31** Most significant bit in the 32-bit integer

The following is an example of how to use the bit fields in an expression:

```
Temperature.08 == 1;
```

The following is an example of how to use the bit fields in a QuickScript:

```
IF Temperature.29 THEN  
    Temperature.29 =0;  
ENDIF;
```

## Tagname Dictionary Utilities

There are two Tagname Dictionary utility programs; DBDump and DBLoad. DBDump is used to export an OpenHMI application Tagname Dictionary as a text file that can be viewed or edited in another program such as Microsoft Excel. DBLoad allows a properly formatted Tagname Dictionary file (created in another program such as Excel or a DBDump file from another OpenHMI application) to be loaded into an existing OpenHMI application. These two programs allow a database (Tagname Dictionary) to be copied, modified or developed in separate portions and then merged into one application.

---

**Note** Both the DBDump and DBLoad utilities are launched from the OpenHMI Application Manager (OPENHMI.EXE). Also, you must convert an application created in an earlier version of OpenHMI before its Tagname Dictionary can be extracted.

---

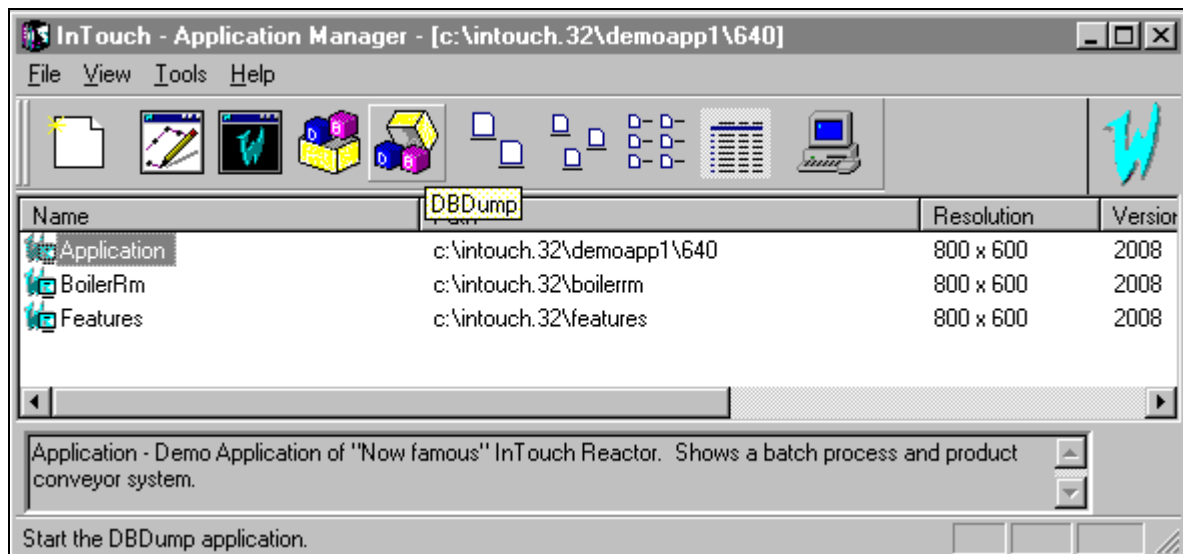
☞ For more information on creating database files, see "Creating a Database Input File."

## DBDump Utility Program

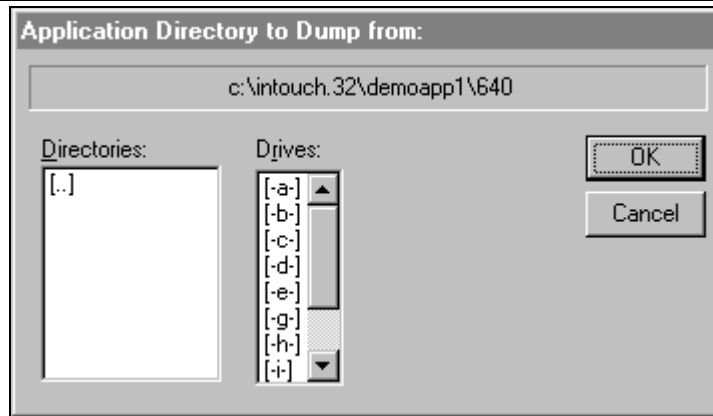


➤ **To extract an existing OpenHMI application's Tagname Dictionary:**

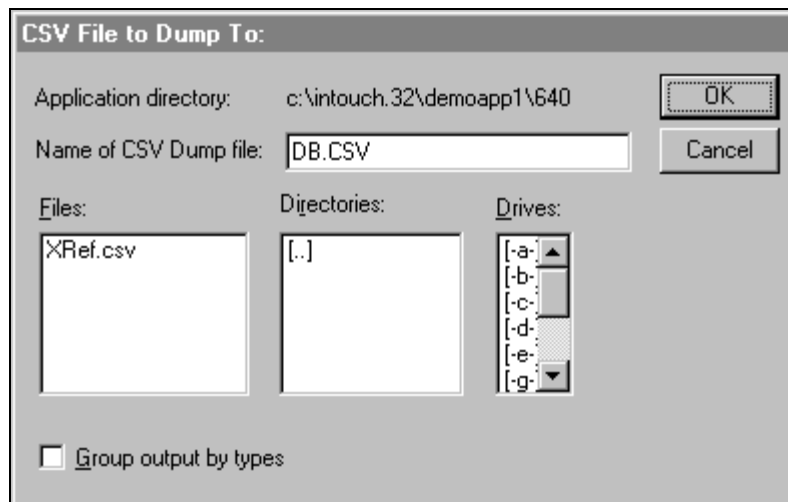
1. Close WindowMaker and WindowViewer if they are running.
2. Start OpenHMI. The **OpenHMI Application Manager** dialog box appears:



3. On the **File** menu, click **DBDump** or, click the DBDump tool. The **Application Directory to Dump from** dialog box appears:



4. Select the OpenHMI application directory into which you want to load the database. (It must replace the default directory shown at the top of the dialog box to be properly selected (in this example **c:\OpenHMI.32**). By default, the last directory you accessed will be displayed when the dialog box initially appears.)
5. Click **OK**. The **CSV File to Dump To:** dialog box appears:



6. In the **Name of CSV Dump file** box, type a name for the file that ends with the .CSV (Comma Separated Variable) extension. (If the name already exists, a message box will appear.)
7. Select **Group output by types** to group the extracted tagnames by tagname type instead of alphabetically by tagname (default.)

---

**Note** In order to extract the database items by groups, the system must read the entire file for each tagname type. Therefore, it takes longer to extract the data. However, when the output file is opened in a .CSV supported application such as Microsoft Excel, the grouping makes it much easier to read or edit. A placeholder for each tagname type is included in the dumped file whether tagnames exist for the type or not.

---

8. Click **OK**. The database information from the selected application directory will be downloaded to the specified filename.

If you open the .CSV file in Microsoft Excel, it sees the comma as a delimiter and automatically separates the data records into columns. For example:



	A	B	C	D	E	F	G	H	I
1	:mode=ask								
2	:DDEAcce	Application	Topic	AdviseActi	DDEProtocol				
3	Server_DD	testprot	topic1	Yes	Yes				
4	Server_IOT	testprot	topic2	Yes	No				
5	:MemoryIn	Group	Comment	Logged	EventLogg	EventLogg	RetentiveV	RetentiveA	Alarm
6	VLoc	\$System		No	No	0	No	No	
7	:MemoryR	Group	Comment	Logged	EventLogg	EventLogg	RetentiveV	RetentiveA	Alarm
8	HLoc	\$System		Yes	No	0	No	No	
9	:MemoryIn	Group	Comment	Logged	EventLogg	EventLogg	RetentiveV	RetentiveA	Alarm
10	Wave1	\$System		Yes	No	0	No	No	
11	Wave2	\$System		Yes	No	0	No	No	
12	Day	\$System		Yes	No	0	No	No	

If you open the .CSV file in Notepad, each data record is separate by a comma. For example:

```

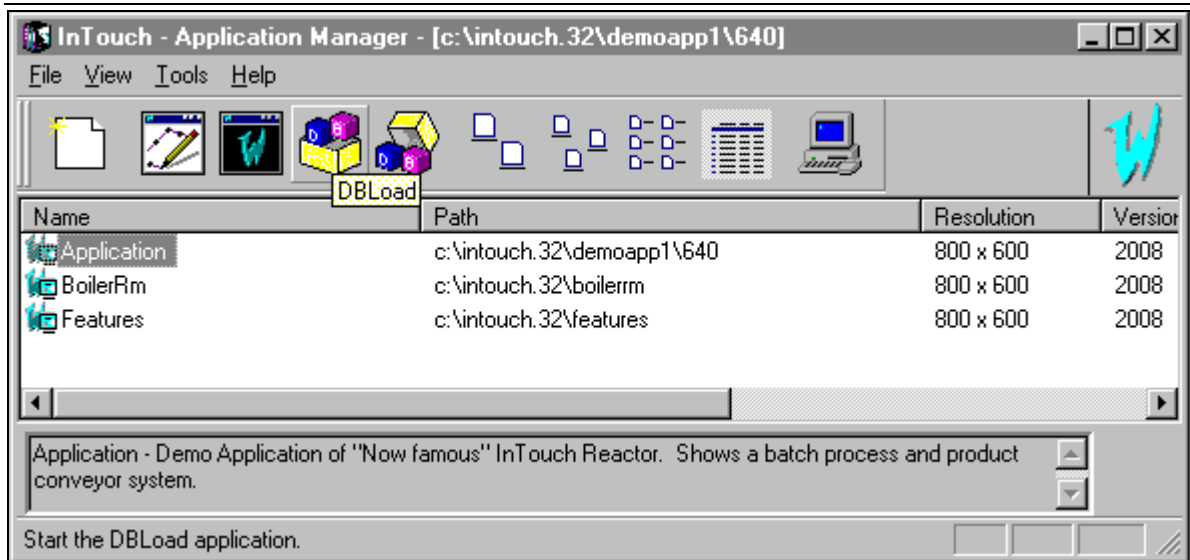
:mode=ask
:DDEAccess,Application,Topic,AdviseActive,DDEProtocol
"Server_DDE","testprot","topic1",Yes,Yes
"Server_IOT","testprot","topic2",Yes,No
:MemoryInt,Group,Comment,Logged,EventLogged,EventLoggingPriority,Retent
"VLoc","$System",,,,,No,No,0,No,No,0,0,,,,0,0,100,0,0,Off,0,1,Off,0,1,0
:MemoryReal,Group,Comment,Logged,EventLogged,EventLoggingPriority,Reten
"HLoc","$System",,,,,Yes,No,0,No,No,0,0,,,,0,0,100,0,0,Off,0,1,Off,0,1,0
:MemoryInt,Group,Comment,Logged,EventLogged,EventLoggingPriority,Retent
"Wave1","$System",,,,,Yes,No,0,No,No,0,0,,,,0,-100,100,0,0,Off,0,1,Off,0
"Wave2","$System",,,,,Yes,No,0,No,No,0,0,,,,0,-100,100,0,0,Off,0,1,Off,0
"Day","$System",,,,,Yes,No,0,No,No,0,0,,,,1,1,31,0,0,Off,0,1,Off,0,1,Off
"OCX_BkgdColor","$System",,,,,No,No,0,No,No,0,0,,,,65535,0,65535,0,0,Off
"currentWindow","$System",,,,,Yes,No,0,No,No,0,0,,,,1,1,15,0,0,Off,0,1,Off
    
```

## DBLoad Utility Program

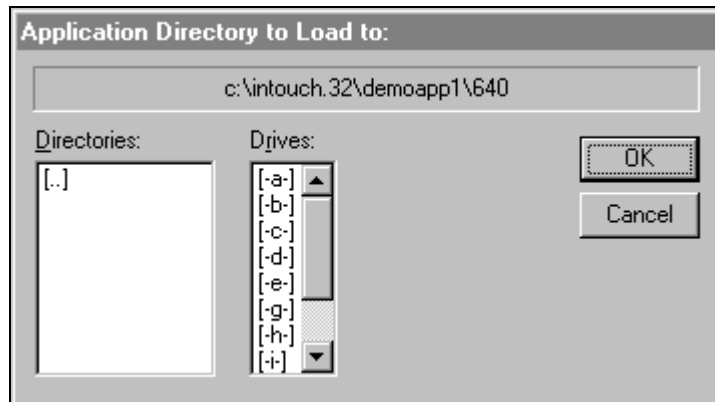


To load/merge a database input file into an existing OpenHMI:

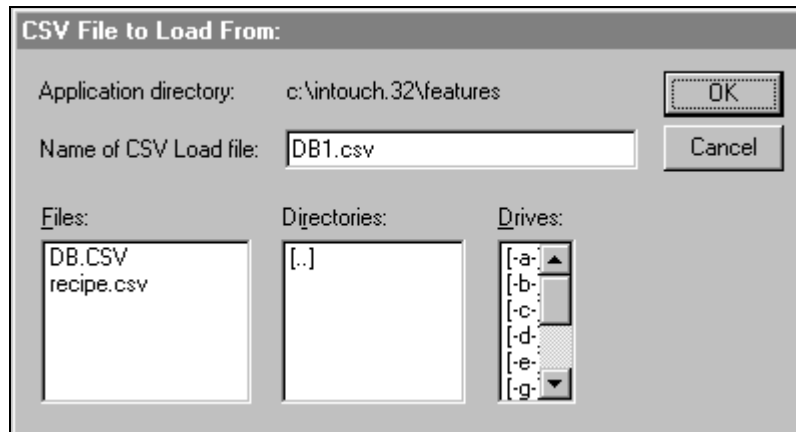
1. Close WindowMaker and WindowViewer if they are running.
2. Start OpenHMI. The **OpenHMI Application Manger** dialog box appears:



3. On the **File** menu, click **DBLoad** or, click the DBLoad tool. The **Application Directory to Load to** dialog box appears:



4. Select the OpenHMI application directory into which you want to load the database. (It must replace the default directory shown at the top of the dialog box to be properly selected (in this example **c:\OpenHMI.32**). By default, the last directory you accessed will be displayed when the dialog box initially appears.)
5. Click **OK**. A message box will appear asking if you have backed up your application. Click **Yes** to continue. The **CSV File to Load From:** dialog box appears:




6. In the **Name of CSV Load file** box, type the path to the .CSV file that you want to load or, locate the file using the **Directories** and **Drive** list boxes. (Once the file is properly selected its name will be the box.)
7. Click **OK**. The database information contained in the selected file will begin uploading to the selected application's Tagname Dictionary.

## Creating a Database Input File


The DBDump and DBLoad database utilities are the tools that you use for performing batch type operations on a Tagname Dictionary. Database input files can be created in any program that supports the comma separated variable file format (.CSV). (The database input file must be created in the comma separated variable format.) For example, WordPad, Notepad and Microsoft Excel. Once an input file is created, the DBLoad program is used to load/merge the data contained in the file into an existing OpenHMI application's database.


You can create a database input file "template" by creating a new OpenHMI application and then running the DBDump program to dump its database to create a correctly formatted .CSV file. This makes entering your modifications easier than creating the input file from scratch.

 For more information, see "Creating Database Record Templates."

## Database Input File Format

The first line of a database input file should specify the operating **:mode** for the file when it is loaded/merged into an application via **DBLoad**.

 If you do not specify a mode, **:mode=test**, by default, **Ask** will be used.


 For more information on valid mode keywords, see "Database Input File Operating Modes."

All data records must begin with the valid keyword for the tagname's **:type**, followed by the valid keyword for each data record (separated by commas):

**:mode=test**

**:IOMsg,Group,Comment,Logged,Event Logged,Event Logging Priority, . . .**

There is a valid **keyword** for each tagname type and data record

 For more information on valid tagname types and the data record entries, "Type and Keyword Entries."

The actual tagname is then entered, followed by the values (separated by commas) for each data record:

**:mode=test**

**:IOMsg,Group,Comment,Logged,EventLogged,EventLoggingPriority, . . .**

**Ingredient\_1,\$System,"",No,No,999, . . .**

In the above example, **IOMsg** = Ingredient\_1, **Group** = \$System, **Logged** = No, **EventLogged** = No, and **EventLoggingPriority** = 999. The data record **Comment** will be blank since "" was entered.

A colon (:) must precede the mode and type keywords. To continue a line, a backslash (\) is entered at the end of the line. Comments may be entered by preceding them with a semi-colon (;).

## Database Input File Operating Modes

The following lists the valid operating mode **keywords** and the action which occurs in each mode when a duplicate tagname is encountered during the load:

**:MODE=REPLACE**

**:MODE=UPDATE**

**:MODE=ASK**

**:MODE=IGNORE**

**:MODE=TERMINATE**

**:MODE=TEST**

**:MODE=REPLACE**

Delete the existing entry and replace it with the new entry.

**:MODE=UPDATE**

Overwrite the existing definition with only the fields that are explicitly defined in the input file.

Fields are considered explicitly defined if the field is in the record and entered by you or is set by the "**:KEYWORD=value**" mechanism. If a field is not specified in the record and the keyword has been reset via the "**:KEYWORD=**" command, the current field value will not be updated.

The following is an example of what will occur when an input file in the update mode is loaded/merged into an application's database via DBLoad:

**:Mode=update**

**:Group=Group1**

**:IODisc,Group,DConversion**

**Tagname1,Group2,**

; Tagname1's Group updated to Group2 only

**Tagname2,,**

; Tagname2's Group updated to Group1 and the DConversion left as is

**Tagname3,,Reverse**

; Tagname3's Group updated to Group1 and the DConversion to "Reverse"

; the following line "resets" the Group field to its default value

**:Group=**

; Data field "Group" is reset to its default value

**Tagname4,,**

; Tagname4 will be left alone

Comments are allowed in the .CSV file. They must be identified with an opening semi-colon (;).

---

**Note** The tagname types must be compatible if the type is being changed and the tagname is in use. Also, a tagname cannot be changed to **ReadOnly=yes** if the tagname is being used on an input link in the application. Because of these restrictions, you should update the target application's tagname use counts before running DBLoad.

---

**:MODE=ASK**

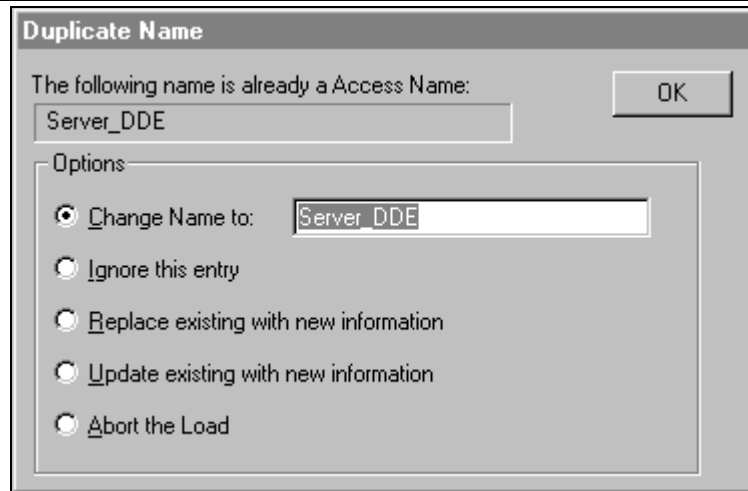
Change the tagname of the input or the existing entry to your specified tagname and then insert the new definition into the Tagname Dictionary.

---

**Note** This is the default input mode if no mode is specified. Each time DBLoad comes across a duplicate tagname, a dialog box will appear asking whether the existing tagname is to be replaced.

---

When the operating mode is set to **ASK**, and a tagname in the input file is a duplicate of a tagname in the target application's Tagname Dictionary, the **Duplicate Name** dialog box will appear:



Select an option, then click **OK**. A confirmation message box will appear when the load is completed.

If a problem occurs that causes the load to fail, a message box will appear. (The error messages are written to the Logger program.)

Option	Description
<b>Change Name to</b>	Replaces the name for the specified tagname with the name typed in the box.
<b>Ignore this entry</b>	Ignores the displayed tagname and processing continues.
<b>Replace existing with new information</b>	Replaces the existing tagname record with the new record.
<b>Update existing with new information</b>	Overwrites the existing tagname record with only the fields that are explicitly defined in the input file.
<b>Abort the Load</b>	Cancels the load function.

### **:MODE=IGNORE**

Ignore the new record and continue processing.

### **:MODE=TERMINATE**

Terminate processing. Do not update target file.

### **:MODE=TEST**

In this mode, DBLoad will act as if it were in the replace mode, but will not modify the database. It will report errors as found in the Logger program and continue with the load. This is useful for verifying the syntax of the input file before actually processing it.

---

**Note** Once the input file is set to the test mode, all other modes are ignored. You can enter `:mode=test` as the first line of the file and not be concerned about other mode changes in the file.

---

*For more information on operating modes, see "Creating a Database Input File."*

# Creating Database Record Templates

**KEYWORDS** can be used to create template records that supply a "global" entry for the respective data fields in subsequent data records. (There is a keyword for each field value that can be set for the tagname except for the **TAGNAME** and **TYPE** fields.)

---

**Note** A template record must follow the formatting requirements previously described for creating a comma separated input file. For example the file must begin with a **:Type** keyword entry.

---

## Setting Field Value Defaults

The keywords can also be used to set the default values for specific fields of a record. For example:

**:KEYWORD=value**

This will set the default value of the referenced field for all subsequent data records. This feature can be used to set the default value for fields that will remain unchanged for a number of records. If a field has a default value defined, the default value is used if there is no data in the record for the value. For example, the result of **:GROUP=Reactor\_Site** would be that all tagnames having a blank entry for the **GROUP** column will be assigned to the **Reactor\_Site** Alarm Group. If the tagname has, for example, **\$System** entered for the **GROUP**, it remains assigned to the Alarm Group **\$System**.

## Resetting Field Value Defaults

The individual keywords can be reset to their original default values by not specifying a value. For example, **:GROUP=**.

## Resetting All Field Value Defaults

To reset all keywords, use the **:RESET** command. This command is entered "as is" with no arguments (**:RESET**) and will affect all entries below the command.

**Note** Default values are the original OpenHMI values for that tagname type. For example, a memory discrete uses the **Group=\$System, EventLogging=Off, InitialValue=Off**, etc. for its default values.

## :Type and Keyword Entries

The table below lists the valid keywords for each tagname **:Type** followed by the keyword for each field value for that type.

<b>:Type &amp; Keywords</b>	<b>Acceptable Values</b>	<b>Default</b>
<b>:MemoryDisc</b>		
AlarmPri	1 to 999	1
AlarmState	None, On, or Off	None
Comment	Any Text String	""
EventLogged	On, Off, Yes, or No	No
EventLoggingPriority	1 to 999	999
Group	Valid Group Name	\$System
InitialDisc	1, 0, On, Off, True, False, Yes, or No	0
Logged	On, Off, Yes, or No	No
OffMsg	Any Text String	None
OnMsg	Any Text String	None
RetentiveValue	1, 0, On, Off, True, False, Yes, or No	No

<b>:Type &amp; Keywords</b>	<b>Acceptable Values</b>	<b>Default</b>
<b>:IODisc</b>		
AlarmPri	1 to 999	1
AlarmState	None, On, or Off	None
Comment	Any Text String	None
DConversion	Direct or Reverse	Direct
AccessName	Valid Access Name	None
EventLogged	On, Off, Yes, or No	No
EventLoggingPriority	1 to 999	999
Group	Valid Group Name	\$\$System
InitialDisc	1, 0, On, Off, True, False, Yes, or No	0
ItemName	Valid Item Name	None
ItemUseTagname	True, False, Yes, or No	Yes
Logged	On, Off, Yes, or No	No
OffMsg	Any Text String	None
OnMsg	Any Text String	None
ReadOnly	Yes or No	No
RetentiveValue	1, 0, On, Off, True, False, Yes, or No	No
<b>:MemoryInt &amp; :Memory Real</b>		
AlarmDevDeadband	Valid Deviation Percentage	0
AlarmValueDeadband	Valid Integer or Real Value	0
Comment	Any Text String	None
Deadband	Valid Integer or Real Value	0
DevTarget	Valid Integer or Real Value	0
EngUnits	Any Text String	None
EventLogged	On, Off, Yes, or No	No
EventLoggingPriority	1 to 999	999
Group	Valid Group Name	\$\$System
HiAlarmPri	1 to 999	1
HiAlarmState	Yes, No, On, or Off	No
HiAlarmValue	Valid Integer or Real Value	0
HiHiAlarmPri	1 to 999	1
HiHiAlarmState	Yes, No, On, or Off	No
HiHiAlarmValue	Valid Integer or Real Value	0
InitialValue	Valid Integer or Real Value	0
LoAlarmPri	1 to 999	1
LoAlarmState	Yes, No, On, or Off	No
LoAlarmValue	Valid Integer or Real Value	0
LogDeadband	Valid Integer or Real Value	0
Logged	On, Off, Yes, or No	No
LoLoAlarmPri	1 to 999	1
LoLoAlarmState	Yes, No, On, or Off	No
LoLoAlarmValue	Valid Integer or Real Value	0
MajorDevAlarmPri	1 to 999	1
MajorDevAlarmState	Yes, No, On, or Off	No
MajorDevAlarmValue	Valid Integer or Real Value	0
MaxValue	Valid Integer or Real Value	9999

<b>:Type &amp; Keywords</b>	<b>Acceptable Values</b>	<b>Default</b>
MinorDevAlarmPri	1 to 999	1
MinorDevAlarmState	Yes, No, On, or Off	No
MinorDevAlarmValue	Valid Integer or Real Value	0
MinValue	Valid Integer or Real Value	0
RetentiveAlarmParameters	On, Off, Yes, or No	No
RetentiveValue	On, Off, Yes, or No	No
ROCAAlarmPri	1 to 999	1
ROCAAlarmState	Yes, No, On, or Off	No
ROCAAlarmValue	Any Valid Percentage	0
ROCTimeBase	Sec, Min, or Hr	Min
<b>:IOInt &amp; :IOReal</b>		
AlarmDevDeadband	Valid Deviation Percentage	0
AlarmValueDeadband	Valid Integer or Real Value	0
Comment	Any Text String	None
Conversion	Linear or Square Root	Linear
AccessName	Valid Access Name	None
Deadband	Valid Integer or Real Value	0
DevTarget	Valid Integer or Real Value	0
EngUnits	Any Text String	None
EventLogged	On, Off, Yes, or No	No
EventLoggingPriority	1 to 999	999
Group	Valid Group Name	\$System
HiAlarmPri	1 to 999	1
HiAlarmState	Yes, No, On, or Off	No
HiAlarmValue	Valid Integer or Real Value	0
HiHiAlarmPri	1 to 999	1
HiHiAlarmState	Yes, No, On, or Off	No
HiHiAlarmValue	Valid Integer or Real Value	0
InitialValue	Valid Integer or Real Value	0
ItemName	Valid Item Name	None
ItemUseTagname	True, False, Yes, or No	No
LoAlarmPri	1 to 999	1
LoAlarmState	Yes, No, On, or Off	No
LoAlarmValue	Valid Integer or Real Value	0
LogDeadband	Valid Integer or Real Value	0
Logged	On, Off, Yes, or No	No
LoLoAlarmPri	1 to 999	1
LoLoAlarmState	Yes, No, On, or Off	No
LoLoAlarmValue	Yes, No, On, or Off	0
MajorDevAlarmPri	1 to 999	1
MajorDevAlarmState	Yes, No, On, or Off	No
MajorDevAlarmValue	Valid Integer or Real Value	0
MaxEU	Valid Integer or Real Value	9999
MaxRaw	Valid Integer or Real Value	9999
MinEU	Valid Integer or Real Value	0
MinorDevAlarmPri	1 to 999	1
MinorDevAlarmState	Yes, No, On, or Off	No
MinorDevAlarmValue	Valid Integer or Real Value	0
MinRaw	Valid Integer or Real Value	0



<b>:Type &amp; Keywords</b>	<b>Acceptable Values</b>	<b>Default</b>
ReadOnly	Yes or No	No
RetentiveAlarmParameters	On, Off, Yes, or No	No
RetentiveValue	On, Off, Yes, or No	No
ROCArmPri	1 to 999	1
ROCArmState	Yes, No, On, or Off	No
ROCArmValue	Any Valid Percentage	0
ROCTimeBase	Sec, Min, or Hr	Min
<b>:MemoryMsg</b>		
Comment	Any Text String	None
EventLogged	On, Off, Yes, or No	No
EventLoggingPriority	1 to 999	999
Group	Valid Group Name	\$\$System
InitialMessage	Any Text String	None
Logged	On, Off, Yes, or No	No
MaxLength	1-131 characters	131
RetentiveValue	On, Off, Yes, or No	No
<b>:IOMsg</b>		
Comment	Any Text String	None
AccessName	Valid Access Name	None
EventLogged	On, Off, Yes, or No	No
EventLoggingPriority	1 to 999	999
Group	Valid Group Name	\$\$System
InitialMessage	Any Text String	None
ItemName	Valid Item Name	None
ItemUseTagname	True, False, Yes, or No	No
Logged	On, Off, Yes, or No	No
MaxLength	1-131 characters	131
ReadOnly	Yes or No	No
RetentiveValue	On, Off, Yes, or No	No
<b>:GroupVar</b>		
Comment	Any Text String	None
Group	Valid Group Name	\$\$System
EventLogged	On, Off, Yes, or No	No
:Type & Keywords	Acceptable Values	Default
<b>:HistoryTrend</b>		
Comment	Any Text String	None
Group	Valid Group Name	\$\$System
<b>:TagID</b>		
Comment	Any Text String	None
Group	Valid Group Name	\$\$System
<b>:IndirectDisc</b>		
Comment	Any Text String	None
EventLogged	On, Off, Yes, or No	No
EventLoggingPriority	1 to 999	999
Group	Valid Group Name	\$\$System
RetentiveValue	On, Off, Yes, or No	No

<b>:Type &amp; Keywords</b>	<b>Acceptable Values</b>	<b>Default</b>
<b>:IndirectAnalog</b>		
Comment	Any Text String	None
EventLogged	On, Off, Yes, or No	No
EventLoggingPriority	1 to 999	999
Group	Valid Group Name	\$\$System
RetentiveValue	On, Off, Yes, or No	No
<b>:IndirectMsg</b>		
Comment	Any Text String	None
EventLogged	On, Off, Yes, or No	No
EventLoggingPriority	1 to 999	999
Group	Valid Group Name	\$\$System
RetentiveValue	On, Off, Yes, or No	No
<b>:Access</b>		
AdviseActive	Yes or No	Yes
NodeName	Valid network node name	
Application	Valid I/O application name	None
Topic	Valid I/O topic name	None
DDEProtocol	Yes specifies the DDE protocol No specifies the SuiteLink protocol	Yes
<b>:AlarmGroup</b>		
Comment	Any Text String	None
EventLogged	On, Off, Yes, or No	No
EventLoggingPriority	1 to 999	999
Group	Valid Group Name	\$\$System

## CHAPTER 4

# Creating Animation Links

Once you create a graphic object or symbol you can "bring it to life" by animating it. By attaching animation links, you can make the object or symbol change in appearance to reflect changes in the value of a tagname or an expression. For example, you can create a pump symbol that is red when it is off and green when it is on. You can also make the pump symbol a touch-sensitive pushbutton that the operator can click with the mouse or touch (when using a touch screen) to turn the pump on and off. You can use these and many other special effects by defining animation links for your objects or symbols.

OpenHMI supports two basic types of links: Touch Links and Display Links. Touch Links allow operator input into the system. Display Links allow output to the operator. Value sliders or pushbuttons are examples of Touch Links. Color fill, location or blink links are examples of Display Links.

This chapter describes the procedures you use to create each type of animation link.

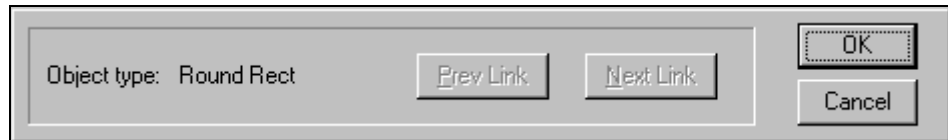
# Common Animation Link Features

Many of the animation links share the following common controls:

- Object Type Dialog Box
- Common Color Palette
- Quick access to the Tag Browser
- Quick access to the Tagname .Fields
- Support for Key Equivalents
- Right-click mouse support in the **Tagname** or **Expression** input boxes (displays a menu with commands that you can apply to the selected text)

## Object Type Dialog Box

The **Object Type** dialog box appears at the top of the screen above the link selection dialog box. It is the header dialog box that is common to all links created. It displays the description of the type of object that you have selected for animation link attachment. For example, **Button**.

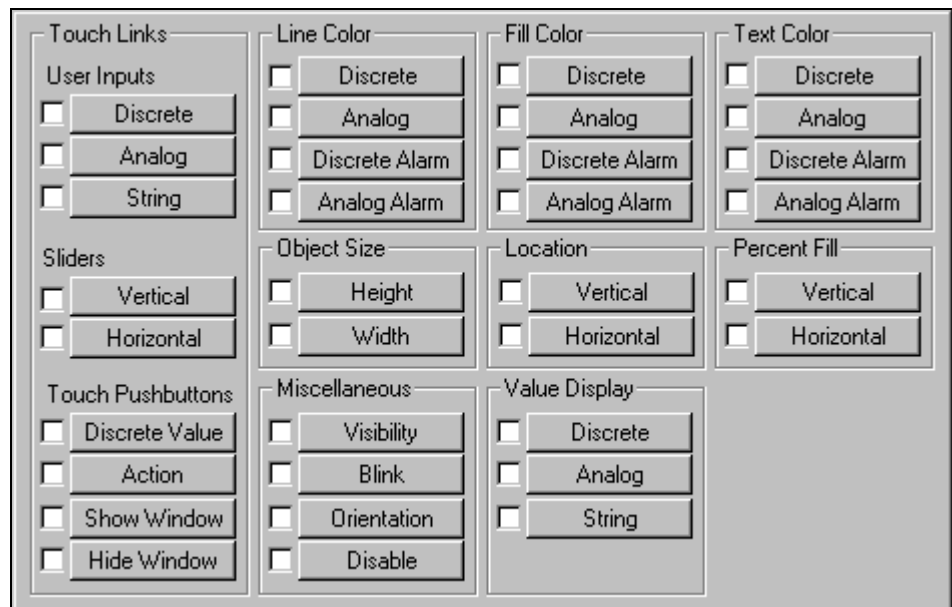


If multiple links have been attached to an object, you can click **Prev Link** and **Next Link** to quickly page forward or backwards through the link dialog boxes for each link attached to the object.

🔗 Links are stored in the order in which they were originally attached to the object.

## Animation Link Selection Dialog Box

You can define multiple links for your objects or symbols. By combining various links, you can create almost any screen animation effect imaginable. You can make objects change color, size, location, visibility, fill level, and so on.



## Assigning Key Equivalents

You can assign a specific key on the keyboard to activate some animation links. The key equivalent is only operational when the object with the link is visible or selected.

If the object has a visibility or disable link, the key equivalent is not active when the object is invisible or disabled.

You can define the same key in multiple windows. However, the definition in the most recently opened window will be the active one. In the case of overlay windows, the key will be active in the window on top.

---

**Note** If any object or action pushbutton in the active window is assigned to the same key used for a **Key Action Script**, the key equivalent link on the key in the active window will take precedence over the execution of the **Key Action Script**.

---

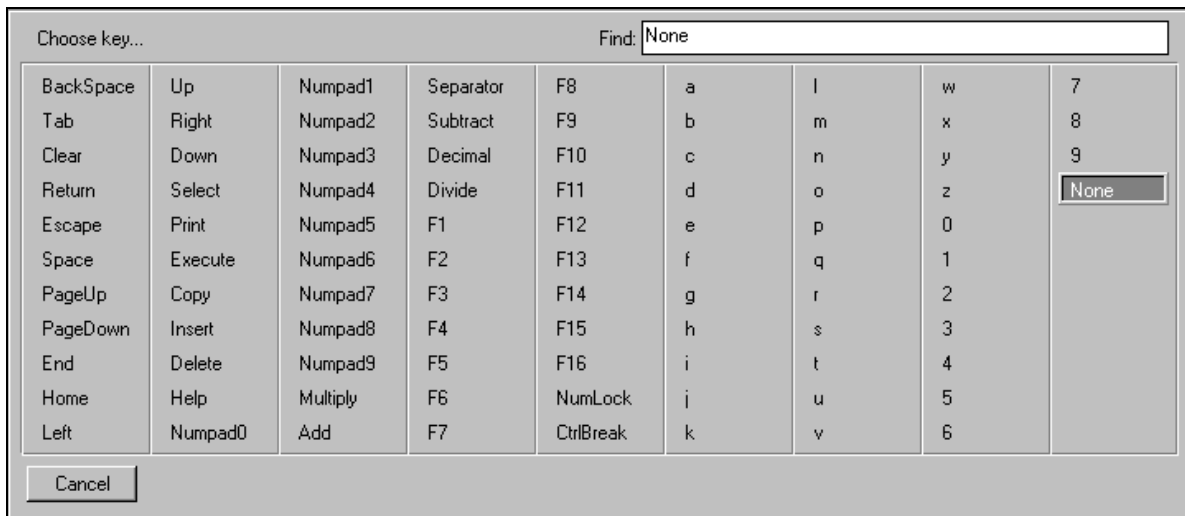
For more information on **Key Action Scripts**, see Chapter 5 - Creating QuickScripts in OpenHMI.

The animation links that support key equivalents will display the **Key Equivalent** group in their link dialog boxes. For example:



➤ **To assign a key to a link:**

1. Select **Ctrl** and/or **Shift** if you want the operator to hold down either or both of these keys when pressing the key equivalent.
2. Click **Key**. The **Choose key** dialog box appears:



3. Click the key that you want to assign to the link. The dialog box will close and the link dialog box will reappear displaying the name of the selected key next to the **Key** button.

---

**Note:** Do not use assign links to the F11-F16 keys. Instead, assign Shift F1 for F11, Shift F2 for F12, etc. These assignment substitutions are required, because the firmware on an OpenHMI workstation interprets F11 to F16 keypad input as Shift F1 to Shift F6 keypad input.

---

## Applying Color Links

You can apply color to the dynamic properties of lines, rectangles, round rectangles, ellipses, polylines, polygons and text. When you create color links for lines, fill or text objects, you will use the color palette to select the colors that you want linked to the value of the tagname, the tagname's alarm state, and so on.

You must use solid colors for line and text color links. If you select a dithered (mixed) color, WindowMaker, by default, will select the closest solid color. You can create custom color palettes and load them into the standard WindowMaker color palette.



When you attach a color link to an object or symbol and you click on a color box in a link's dialog box, the color palette appears.

Click the color you want to use for the link. The color palette automatically closes and the color you selected will appear in the color box in the link detail dialog box.


For more information on customizing the color palette, see Chapter 1 - WindowMaker Program Elements.

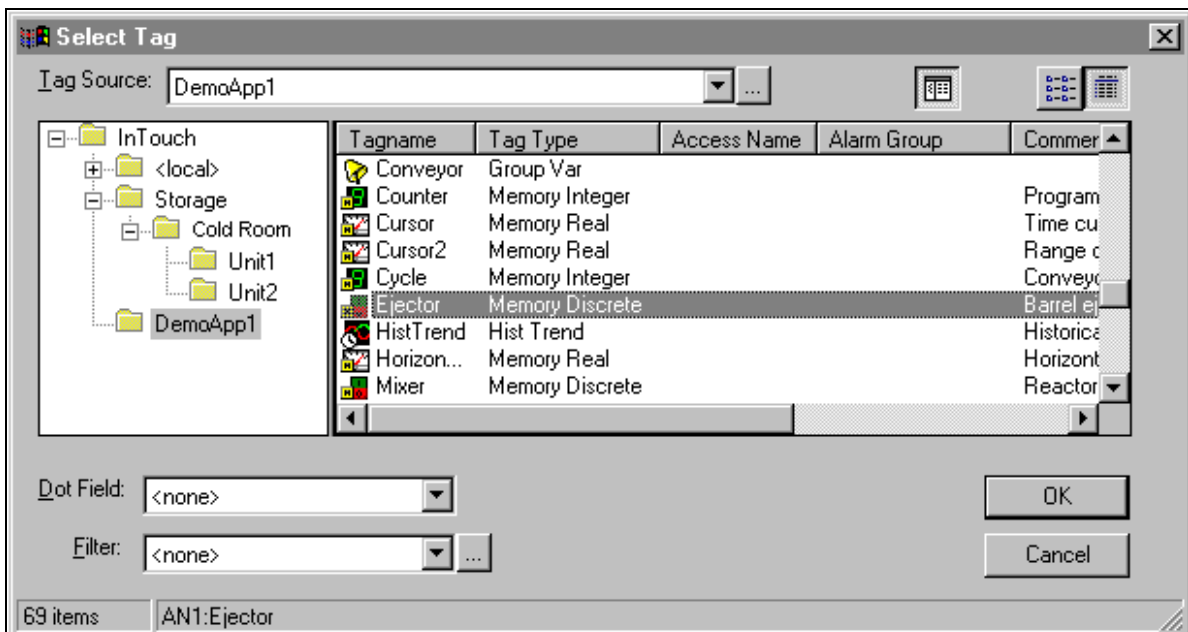
## Accessing the Tag Browser

You can quickly view all of the tagnames that are defined in your application when you are creating animation links by accessing the Tag Browser. If you select the tagname that you want assigned to your link from the Tag Browser, it is automatically inserted into the **Tagname** or **Expression** box.

➤ **To access the Tag Browser:**

1. Double-click any blank animation link **Tagname** or **Expression** input box. The Tag Browser will appear.

2. Click the  tool to show the tree view pane displaying all defined tag sources:



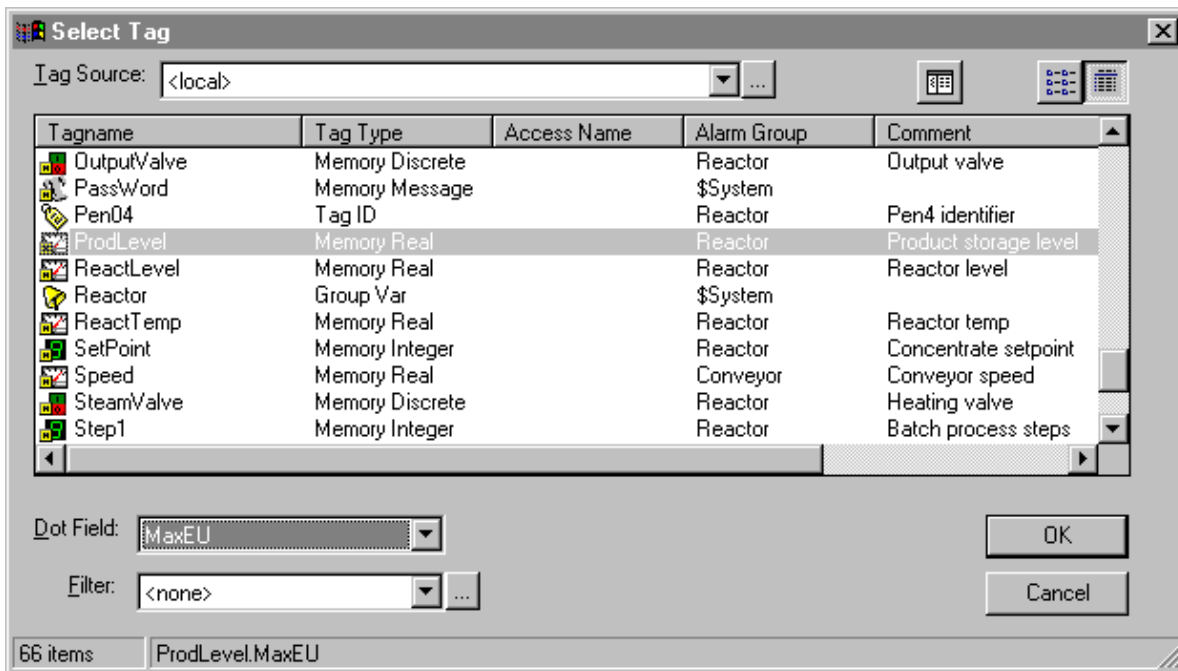
- ☞ If you are not using the tree view mode, click the **Tag Source** arrow and select the name for the tag source that you want to display in the list. The Tag Browser will refresh and the selected tag source's tagnames will be displayed.
- 3. Select the tagname you want to use for the link then click **OK** or, double-click the tagname to simultaneously select it, close the Tag Browser and insert it into the **Tagname** or **Expression** box.
- ☞ To use a **.field** with the selected tagname, click the **Dot Field** arrow and select the **.field** that you want to use in the list, and then click **OK**.  
 To display a tagname's database definition, type the tagname in the **Tagname** or **Expression** box, then double-click it. The **Tagname Dictionary** dialog box will appear displaying the tagname's definition.
- ☞ For more information on the Tag Browser, see Chapter 3 - Tagname Dictionary.

## Accessing Tagname .Fields

There are two methods that you can use to access tagname **.fields** from an animation link **Tagname** or **Expression** input box. The two methods are described below.

➤ **To access tagname .fields through the Tag Browser:**

1. Double-click a blank **Tagname** or **Expression** input box. The Tag Browser appears displaying the tagnames defined for the current tag source:



2. Click the **Dot Field** arrow to open the list of **.fields** that you can associate with the type of tagname currently selected.
  - ☞ By default, **<none>** will initially be displayed for all types of tagnames.
3. Click the **.field** in the list that you want to append to the selected tagname.

**Note** Not every tagname type has the same **.fields**. For example, a **Discrete** type tagname has **.OnMessage**, whereas an analog does not. If you select a **Discrete** type tagname and you assign **.OnMessage** to it, and then you select another **Discrete** type tagname, the displayed **.field** list will not change. But, if you select

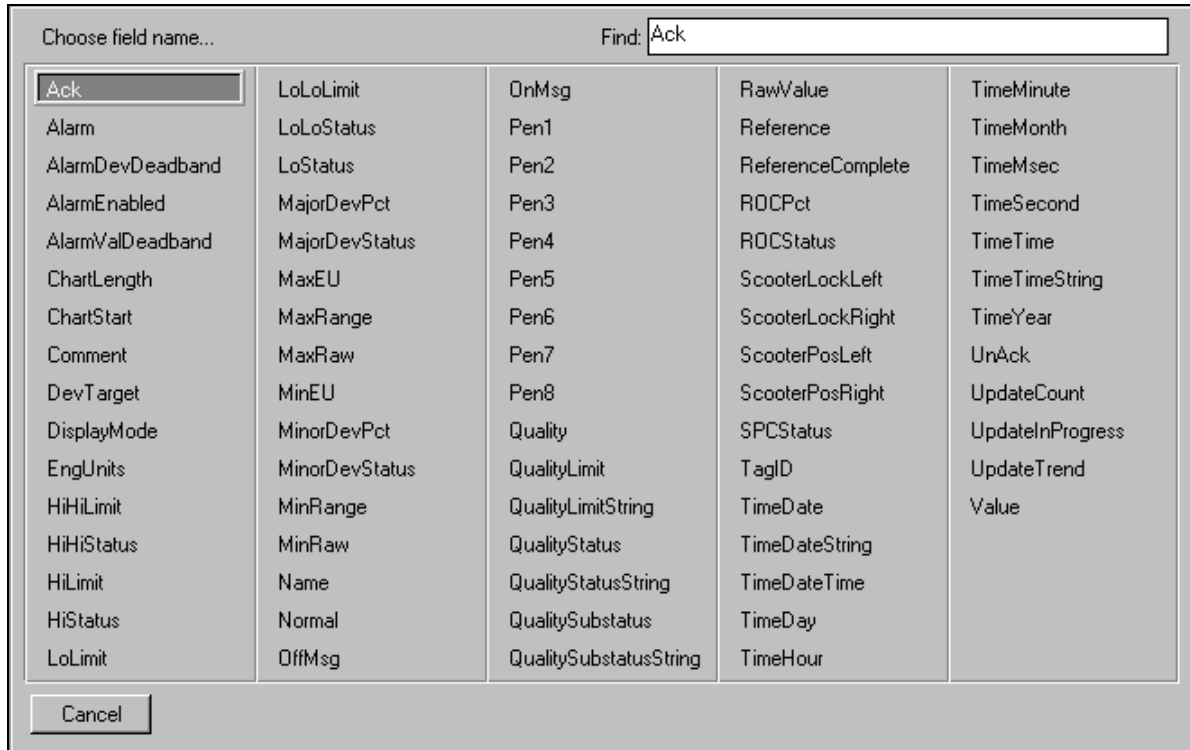
another type of tagname in the control view list, for example an analog, the displayed **.field** will revert to **<none>**.

☞ For more information on the Tag Browser, see Chapter 3 - Tagname Dictionary.

📖 For more information on tagname **.fields**, see your *OpenHMI Reference Guide*.

➤ **To access the tagname .fields through the Choose field name dialog box:**

1. In **Tagname** or **Expression** input box, type a tagname plus a period (**tagname.**) and then double-click to the right of it, or type just a period, and then double-click to the right of it. The **Choose field name** dialog box appears displaying all tagname **.fields**:



2. Select the **.field** that you want to use. The dialog box will close and the selected **.field** will automatically be inserted into the **Tagname** or **Expression** input box.

## Animating Objects

➤ **To animate an object or symbol:**

1. Create and select the object (line, filled shape, text, button or symbol) that you want to animate.
2. On the **Special** menu, click **Animation Links** or, double-click on the object. The dialog box containing all animation links appears:
  - ☞ You can also right-click the object, and then click **Animation Links**.



- Click the button for the link that you want to attach to the selected object.

☞ If a link is not applicable for the selected object, its button will not be active.

Clicking the check box only selects the link. Clicking the link name button selects the link and opens its detail definition dialog box. The check box will automatically be selected when you click the link name button and accept the input. However, if you clear a link's check box, the animation link is removed from the selected object.

---

**Note** You will not be able to modify the definition of the default link unless you click the button.

---

- Enter the details for the link definition, and then click **OK**. The **Link Selection** dialog box will reappear and if desired, you can create another link for the object.
- Click **OK** to accept all links for the object and close the dialog box.

☞ When you are creating animation links, the tagname you type in the animation link's tagname or expression box must be defined in the Tagname Dictionary before the link can be assigned to it. If it is not defined, a message box will appear asking you if you want to define the tagname now. If you click **Yes**, the Tagname Dictionary will appear and you can define the tagname.

## Creating Touch Links

You use **Touch Links** on objects or symbols that you want to be "touch-sensitive" in runtime. They allow the operator to input data into the system. For example, the operator may turn a valve on or off, enter a new alarm setpoint, run a complex logic script or log on using text strings, and so on.

Touch links are easily identified in runtime because a "frame" surrounds a touch-sensitive object when you pass the cursor over it or you press the TAB key to move from object to object. If a touch link object or symbol contains text objects that are placed on top of each other, the top text object will be used to display the data value.

The operator activates a touch-sensitive pushbutton by clicking it, touching the object (when using a touch screen), pressing an assigned key equivalent or, pressing the ENTER key when the object is "framed."

There are nine types of touch links that you can create:

Touch Link	Types
<b>User Inputs</b>	Discrete, Analog, String
<b>Sliders</b>	Vertical, Horizontal
<b>Touch Pushbuttons</b>	Discrete Value, Action, Show Window, Hide Window

The following sections describe how to create each of the touch links.

---

**Note** If the object or symbol used for these links (with the exception of 3-D buttons) contains a text field, all attributes currently set for text, (justification, style, font, and so on) will be applied when the text object is displayed in WindowViewer. When a text field is used for inputting the value, the output value will also be displayed unless the **Input Only** option in the respective tagname's detail dialog box is enabled.

---

## Creating User Input Touch Links

You use **User Input Touch Links** to create touch-sensitive objects that allow operator input into the system. For example, pushbuttons to change discrete states, analog values or security log-ons. There are three types of **User Input** touch links:

User Input	Description
<b>Discrete</b>	Used to control the value of a discrete tagname. When this link is activated, a dialog box will appear prompting the operator to make a selection.
<b>Analog</b>	Used to input the value of an analog (integer or real) tagname. When the link is activated, an input box will appear and the value may be entered from the standard keyboard or an optional on-screen keypad.
<b>String</b>	Used to create an object into which a string message may be input. When the link is activated, an input box will appear for entering the message value or an optional on screen keyboard.

➤ **To create a discrete input link:**

1. Double-click the object or select it, and then on the **Special** menu, click **Animation Links**. The link selection dialog box will appear.
  - ☞ To quickly access the animation link selection dialog box, right-click the object, and then click **Animation Links**.
2. In the **User Inputs** section, click **Discrete**. The **Input -> Discrete Tagname** dialog box appears:

3. In the **Tagname** box, type a discrete type tagname.
  - ☞ Right-click the **Tagname** box, to access the commands that you can apply to the selected text.
4. Click **Key** if you want to assign a key equivalent to the link.
  - ☞ For more information on assigning keys, "Assigning Key Equivalents."
5. In the **Msg to User** box, type the message that you want to appear in the input dialog box when the input link is activated.
6. In the **Set Prompt** and **Reset Prompt** boxes, type the messages that you want to display on the buttons the operator will click in the input dialog box to turn the discrete value on and off.
7. In the **On Message** and **Off Message** boxes, type the messages that you want to appear in the text field (if any) associated with the object when the object is on or off.
8. Select **Input Only** if you want to prevent the input from being displayed in a text field associated with the object. (This option only applies to an object that has a text field associated with it.)
9. Click **OK** to attach the link to the object and return to the animation links dialog box. You can now attach another link to the object if desired.

---

**Note** If the tagname you entered is not defined in your Tagname Dictionary you will be prompted to define it now.

---

➤ **To create an analog input link:**

1. Double-click the object or select it, and then on the **Special** menu, click **Animation Links**. The link selection dialog box will appear.
  - ☞ To quickly access the animation link selection dialog box, right-click the object, and then click **Animation Links**.
2. In the **User Inputs** section, click **Analog**. The **Input -> Analog Tagname** dialog box appears:

**Note** If a text field is being used for this link, it must be formatted properly in order to display the analog (integer or real) value's output correctly.

For more information on formatting text fields, see Chapter 2 - Using WindowMaker.

3. In the **Tagname** box, type an analog (integer or real) type tagname.
  - Right-click the **Tagname** box, to access the commands that you can apply to the selected text.
4. Click **Key** if you want to assign a key equivalent to the link.
  - For more information on assigning keys, see "Assigning Key Equivalents."
5. If you are displaying the optional keypad when this link is activated, in the **Msg to User** box, type the prompt message that you want to appear in keypad.
6. If you want to display an on-screen numeric keypad for inputting the new value of the string, select **Yes**.
7. In the **Min Value** box, type the minimum input value for the tagname.
8. In the **Max Value** box, type the maximum input value for the tagname.
9. Select **Input Only** if you want to prevent the input from being displayed in a text field associated with the object. (This option only applies to an object that has a text field associated with it such as a 3 dimensional button.)
10. Click **OK** to attach the link to the object and return to the animation links dialog box. You can now attach another link to the object if desired.

**Note** If the tagname you entered is not defined in your Tagname Dictionary you will be prompted to define it now.

➤ **To create a string input link:**

1. Double-click the object or select it, and then on the **Special** menu, click **Animation Links**. The link selection dialog box will appear.
  - To quickly access the animation link selection dialog box, right-click the object, and then click **Animation Links**.
2. In the **User Inputs** section, click **String**. The **Input -> String Tagname** dialog box appears:

3. In the **Tagname** box, type a message type tagname.
  - ☞ Right-click the **Tagname** box, to access the commands that you can apply to the selected text.
4. Click **Key** if you want to assign a key equivalent to the link.
  - ☞ For more information on assigning keys, see "Assigning Key Equivalents."
5. If you are displaying the optional keypad when this link is activated, in the **Msg to User** box, type the prompt message that you want to appear in keyboard.
6. If you want the input string to appear on the screen as it is typed, select **Yes** for the **Echo Characters?** option. If the data is sensitive (for example, a password) and should not be visible on the screen, select **No**.
7. If you want to display an on-screen keyboard for inputting the new value of the string, select **Yes** for the **Keypad?** option.
8. Select **Input Only** if you want to prevent the input from being displayed in a text field associated with the object. (This option only applies to an object that has a text field associated with it such as a 3 dimensional button.)

---

**Note** When WindowViewer is initially started, the string will display the text you typed in the **Initial Value** box when you defined the tagname linked to this object.

---

9. Click **OK** to attach the link to the object and return to the animation links dialog box. You can now attach another link to the object if desired.

---

**Note** If the tagname you entered is not defined in your Tagname Dictionary you will be prompted to define it now.

---

## Creating Slider Touch Links

You use **Slider Touch Links** to create objects or symbols that can be moved around the window with the mouse or other pointing devices such as a finger on a touch screen. As the object or symbol is moved, it alters the value of the tagname linked to it. This provides the ability to create devices for setting values in the system.

An object may have a horizontal or a vertical slider touch link, or both. By using both links on a single object, the value of two analog tagnames can be altered simultaneously.

---

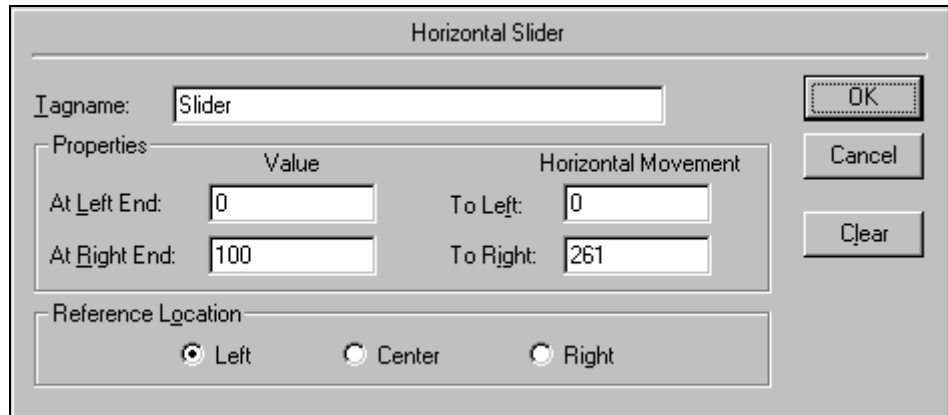
**Note** The horizontal and vertical slider links are created the same way. This procedure describes the **Horizontal Slider** link.

---

- **To create a horizontal (or vertical) slider link:**
  1. Double-click the object or select it, and then on the **Special** menu, click **Animation Links**. The link selection dialog box will appear.

☞ To quickly access the animation link selection dialog box, right-click the object, and then click **Animation Links**.

2. In the **Slider** section, click **Horizontal**. The **Horizontal Slider** dialog box appears:



3. In the **Tagname** box, type an analog (integer or real) type tagname.

☞ Right-click the **Tagname** box, to access the commands that you can apply to the selected text.

4. In the **At Left End** box, type the value for the tagname when the slider is in its farthest left position.

5. In the **At Right End** box, type the value for the tagname when the slider is in its farthest right position.

6. In the **To Left** box, type the number of pixels the slider can move to the left.

☞ At the far left position, the tagname's value will be equal to the value entered in the **At Left End** field.

7. In the **To Right** box, type the number of pixels the slider can move to the right.

☞ At the far right position, the tagname's value will be equal to the value entered in the **At Right End** field.

8. Select the **Reference Location** on the object that the cursor will lock on for moving the object.

9. Click **OK** to attach the link to the object and return to the animation links dialog box. You can now attach another link to the object if desired.

---

**Note** If the tagname you entered is not defined in your Tagname Dictionary you will be prompted to define it now.

---

## Creating Touch Pushbuttons Touch Links

You use **Touch Pushbutton Touch Links** to create object links that immediately perform an operation when clicked with the mouse or touched (when touch screen is being used). These operations can be **Discrete Value Changes**, **Action Script** executions, **Show** or **Hide Window** commands. There are four types of **Touch Pushbutton** links:

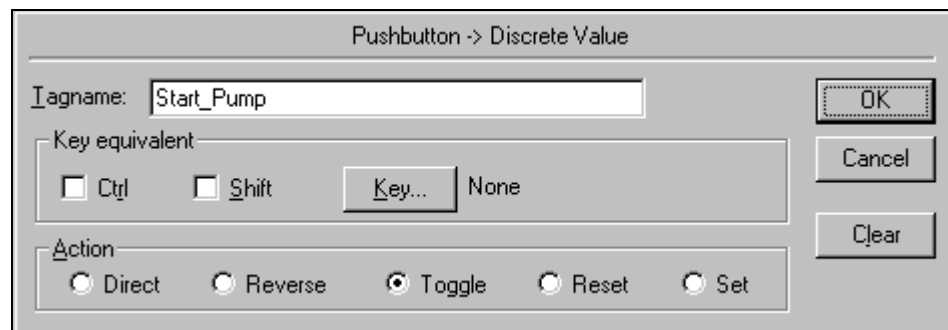
Touch Pushbuttons	Description
<b>Discrete Value</b>	Used to make any object or symbol into a pushbutton that controls the state of a discrete tagname. Pushbutton actions can be set, reset, toggle, momentary on (direct) and momentary off (reverse) types.
<b>Action</b>	Allows any object, symbol or button to have up to three different action scripts linked to it; <b>On Down</b> , <b>While Down</b> and <b>On Up</b> .

Action scripts can be used to set tagnames to specific values, show and/or hide windows, start and control other applications, execute functions, and so on.

- Show Window** Used to make an object or symbol into a button that opens one or more windows when it is clicked or touched.
- Hide Window** Used to make an object or symbol into a button that closes one or more windows when it is clicked or touched.

➤ **To create a discrete value touch pushbutton link:**

1. Double-click the object or select it, and then on the **Special** menu, click **Animation Links**. The link selection dialog box will appear.
  - ☞ To quickly access the animation link selection dialog box, right-click the object, and then click **Animation Links**.
2. In the **Touch Pushbutton** section, click **Discrete Value**. The **Pushbutton -> Discrete Value** dialog box appears:



3. In the **Tagname** box, type a discrete type tagname.
  - ☞ Right-click the **Tagname** box, to access the commands that you can apply to the selected text.
4. Click **Key** if you want to assign a key equivalent to the link.
5. For more information on assigning keys, see "Assigning a Key to an Animation Link."
6. Select the **Action** option that you want to use for the pushbutton as follows:
  - Direct** Sets the value equal to 1 (True, On, Yes) as long as the pushbutton is pressed and held down. The value automatically resets to 0 (False, Off, No) when the button is released.
  - Reverse** Sets the value equal to 0 (False, Off, No) when the pushbutton is pressed and held down. The value automatically resets to 1 (True, On, Yes) when the button is released.
  - Toggle** Reverses the state of the discrete tagname when it is pressed, e.g., if the tagname is equal to 1 and the button is pressed, it is reset to 0 and vice-versa.
  - Reset** Sets the value equal to 0 (False, Off, No) when the pushbutton is pressed.
  - Set** Sets the value equal to 1 (True, On, Yes) when the pushbutton is pressed.
7. Click **OK** to attach the link to the object and return to the animation links dialog box. You can now attach another link to the object if desired.

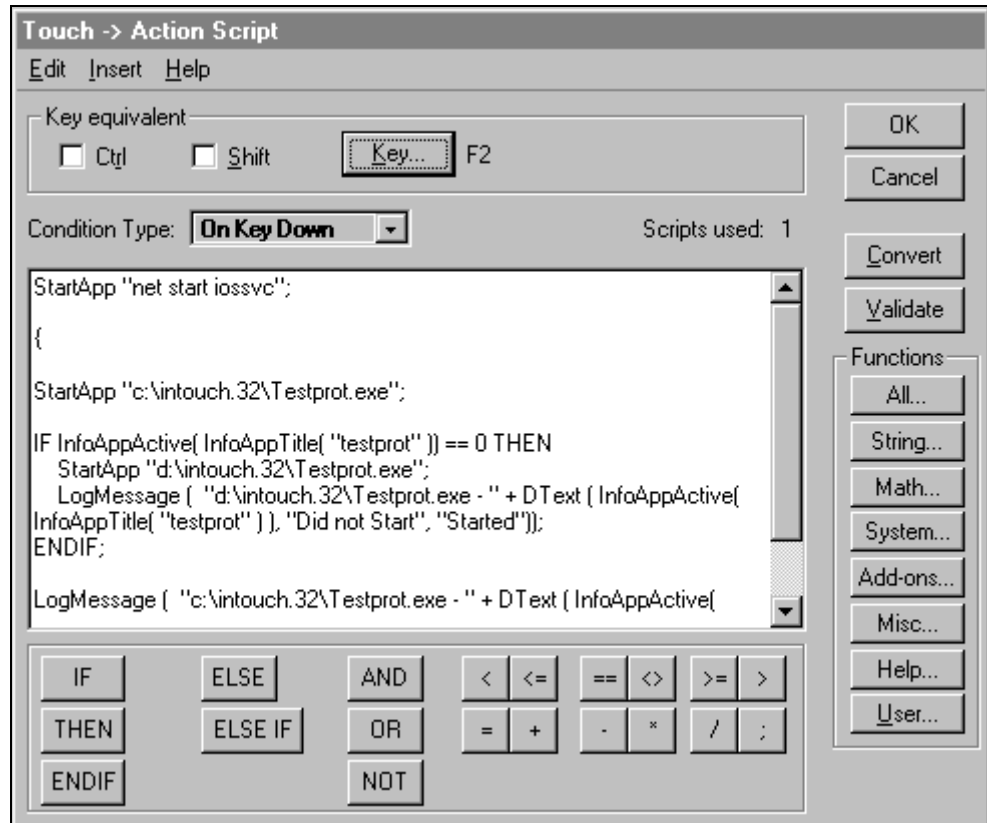
---

**Note** If the tagname you entered is not defined in your Tagname Dictionary you will be prompted to define it now.

---

➤ **To create an action touch pushbutton link**

1. Double-click the object or select it, and then on the **Special** menu, click **Animation Links**. The link selection dialog box will appear.
  - ☞ To quickly access the animation link selection dialog box, right-click the object, and then click **Animation Links**.
2. In the **Touch Pushbutton** section, click **Action**. The **OpenHMI -> Action Script** editor appears:



☞ For more information on writing QuickScripts, see see Chapter 5 - Creating QuickScripts in OpenHMI.

3. Click the **Condition Type** arrow and select the script type that you want to apply to the object. You can apply all three script types to the same key:

**On Key Down** Executes the script one time when the key is initially pressed.

**While Down** Executes the script continuously on a time interval as long as the key is held down.

**On Key Up** Executes the script one time when the key is released.

☞ A **While Down** script will begin executing after the specified number of milliseconds has elapsed. To cause immediate execution, create a duplicate **On Key Down** script.

---

**Note** If any object or action pushbutton in the active window is assigned to the same key used for a **Key** Script, the key equivalent link on the key in the active window will take precedence over the execution of the **Key** script.

---

☞ For more information on scripts, see Chapter 5 - Creating QuickScripts in OpenHMI.

4. Click in the script editor's window and type the script that you want executed when the object is activated.



- 
5. Click **OK** to attach the script to the object and return to the animation links dialog box. You can now attach another link to the object if desired.

---

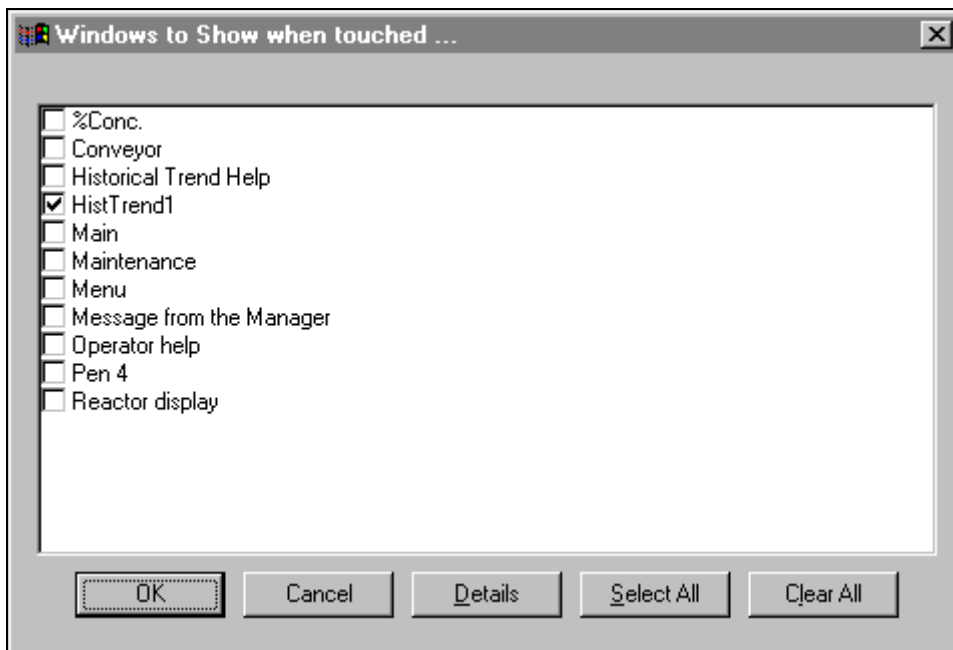
**Note** If the tagname you entered is not defined in your Tagname Dictionary you will be prompted to define it now.

---

➤ **To create a show (or hide) window touch pushbutton link:**

**Note** The **Show Window** and **Hide Window** links are created in the same way. This procedure describes the **Show Window** link.

1. Double-click the object or select it, on the **Special** menu, click **Animation Links**. The link selection dialog box will appear.
  - ☞ To quickly access the animation link selection dialog box, right-click the object, and then click **Animation Links**.
2. In the **Touch Pushbutton** section, click **Show Window**. The **Windows to Show when touched** dialog box appears:



3. Select the windows that you want to open when the object is clicked or touched.
4. Click **OK** to attach the link to the object and return to the animation links dialog box. You can now attach another link to the object if desired.

☞ When showing more than one window where one of the windows is a **Replace** type, if it intersects any other windows, they will close before they are displayed (giving the illusion that your **Show Window** animation link is not working).

To change a window's type, right-click a blank area of the open window, and then click **Window Properties**. The **Window Properties** dialog box will appear and you can change the type. (You cannot change the window's type if WindowViewer is running.)

☞ For more information on window properties, see Chapter 2 - Using WindowMaker.

## Creating Display Links

You use the various **Display Links** to provide output to the operator. There are eight types of display links that you can create:

Display Link	Types
<b>Line, Fill &amp; Text Color</b>	Discrete, Analog, Discrete Alarm, Analog Alarm
<b>Object Size</b>	Height, Width
<b>Location</b>	Horizontal, Vertical
<b>Percent Fill</b>	Horizontal, Vertical
<b>Miscellaneous</b>	Visibility, Orientation, Blink, Disable
<b>Value Display</b>	Discrete, Analog, String

The following sections describe how to create each of the display links.

## Creating Color Links

You use color links to animate the **Line Color**, **Fill Color**, and **Text Color** attributes of an object.

---

**Note** You must use solid colors for line and text color links. If you select a dithered (mixed) color, WindowMaker, by default, will select the closest solid color. To avoid dithered colors, your video card must have at least 2MB and your color depth settings must be higher than 256 colors, such as 32K or 65K (sometimes called "high color".)

---

 For more information on the color palette, see "Applying Color Links."

Each of these color attributes may be made dynamic by defining a color link for the attribute. The color attribute may be linked to the value of a discrete expression, analog expression, discrete alarm status or analog alarm status. There are four types of line, fill and text color:

Color Link	Description
<b>Discrete</b>	Used to control the fill, line and text colors attributes of an object or symbol that is linked to the value of a discrete expression.
<b>Analog</b>	The line, fill, and text color of an object or symbol can be linked to the value of an analog tagname (integer or real) or an analog expression. Five value ranges are defined by specifying four breakpoints. Five different colors can be selected which will be displayed as the value range changes.
<b>Discrete Alarm</b>	The text, line, and fill color of an object can all be linked to the alarm state of a tagname, Alarm Group, or Group Variable. This color link allows a choice of two colors; one for the normal state and one for the alarm state of the tagname. This link can be used for both analog and discrete tagnames. If it is used with an analog tagname, it responds to any alarm condition of the tagname.
<b>Analog Alarm</b>	The text, line, and fill color of an object can all be linked to the alarm state of an analog tagname, Alarm Group, or Group Variable. Allows a specific color to be set for the normal state as well as a separate color for each alarm condition defined for the tagname.

➤ **To create a discrete fill color link:**

---

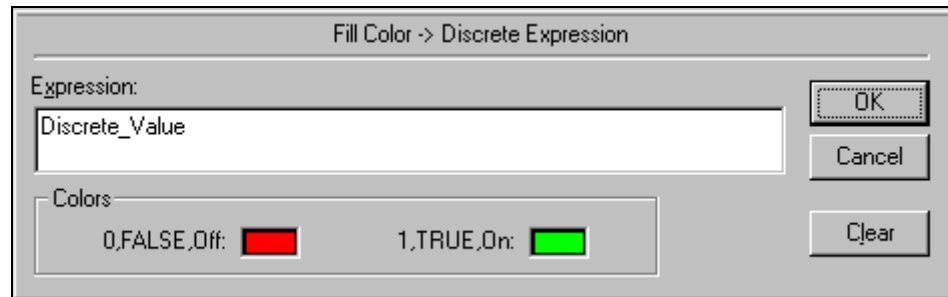
**Note** All of the **Line Color** and **Text Color** links are created in the same way as the **Fill Color** links. The following procedure describes creating a **Fill Color** link.

---

1. Double-click the object or select it, and then on the **Special** menu, click **Animation Links**. The link selection dialog box will appear.

☞ To quickly access the animation link selection dialog box, right-click the object, and then click **Animation Links**.

- In the **Fill Color (Line Color or Text Color)** section, click **Discrete**. The **Fill Color -> Discrete Expression** dialog box appears:



- In the **Expression** box, type a discrete tagname or an expression that equates to true or false.

☞ Up to 256 characters may be typed for your expression.

Discrete expressions can also contain analog tagnames. For example, TankLevel >= 75. In this example, when the value of the variable "TankLevel" is greater than or equal to "75," the fill color of the object will change.

Right-click the **Expression** box, to access the commands that you can apply to the selected text.

- In the **Colors** group, click each color box to open the color palette. Click the color in the color palette that you want to use for each tagname state.

---

**Note** You must use solid colors for line and text color links. If you select a dithered (mixed) color, WindowMaker, by default, will select the closest solid color. To avoid dithered colors, your video card must have at least 2MB and your color depth settings must be higher than 256 colors, such as 32K or 65K (sometimes called "high color".)

---

☞ For more information on the color palette, see "Applying Color Links."

- Click **OK** to attach the link to the object and return to the animation links dialog box. You can now attach another link to the object if desired.

---

**Note** If the tagname you entered is not defined in your Tagname Dictionary you will be prompted to define it now.

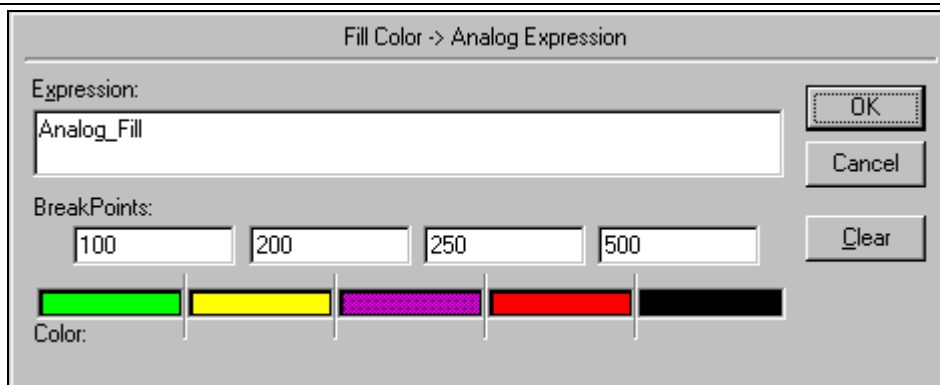
---

➤ **To create an analog expression color link:**

- Double-click the object or select it, and then on the **Special** menu, click **Animation Links**. The link selection dialog box will appear.

☞ To quickly access the animation link selection dialog box, right-click the object, and then click **Animation Links**.

- In the **Fill Color (Line Color or Text Color)** section, click **Analog**. The **Fill Color -> Analog Expression** dialog box appears:



3. In the **Expression** box, type an analog (integer or real) tagname or an expression that equates to an analog value.
  - ☞ Up to 256 characters may be typed for your expression.
  - Right-click the **Expression** box, to access the commands that you can apply to the selected text.
4. In each **Break Points** box, you can specify the breakpoint values (decimals are valid for real type tagnames) where the object will change color.
  - ☞ You do not have to use four different values. For example, if you only want the object to change color three times, type three values then use the same color for the third and fourth values.
5. In the **Colors** group, click each color box to open the color palette. Click the color in the color palette that you want to use for each breakpoint.

---

**Note** You must use solid colors for line and text color links. If you select a dithered (mixed) color, WindowMaker, by default, will select the closest solid color. To avoid dithered colors, your video card must have at least 2MB and your color depth settings must be higher than 256 colors, such as 32K or 65K (sometimes called "high color".)

---

☞ For more information on the color palette, see "Applying Color Links."

6. Click **OK** to attach the link to the object and return to the animation links dialog box. You can now attach another link to the object if desired.

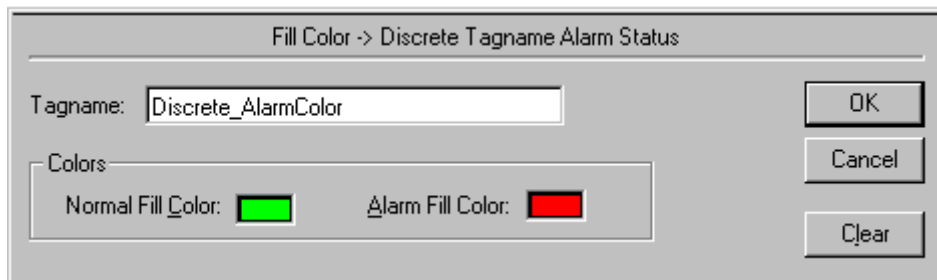
---

**Note** If the tagname you entered is not defined in your Tagname Dictionary you will be prompted to define it now.

---

➤ **To create a discrete alarm status color link:**

1. Double-click the object or select it, and then on the **Special** menu, click **Animation Links**. The link selection dialog box will appear.
  - ☞ To quickly access the animation link selection dialog box, right-click the object, and then click **Animation Links**.
2. In the **Fill Color (Line Color or Text Color)** section, click **Discrete Alarm**. The **Fill Color -> Discrete Tagname Alarm Status** dialog box appears:



3. In the **Tagname** box, type the discrete tagname whose alarm status you want associated with the object.
  - ☞ Right-click the **Tagname** box, to access the commands that you can apply to the selected text.
4. In the **Colors** group, click each color box to open the color palette. Click the color in the color palette that you want to use for each color state.

---

**Note** You must use solid colors for line and text color links. If you select a dithered (mixed) color, WindowMaker, by default, will select the closest solid color. To avoid dithered colors, your video card must have at least 2MB and your color depth settings must be higher than 256 colors, such as 32K or 65K (sometimes called "high color".)

---

☞ For more information on the color palette, see "Applying Color Links."

5. Click **OK** to attach the link to the object and return to the animation links dialog box. You can now attach another link to the object if desired.

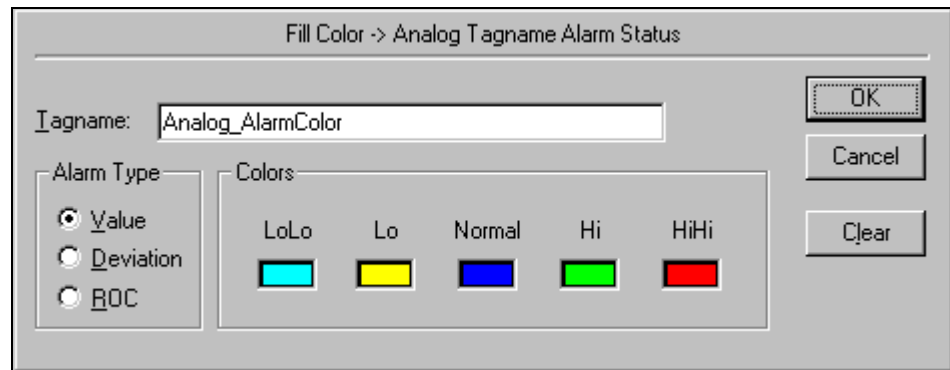
---

**Note** If the tagname you entered is not defined in your Tagname Dictionary you will be prompted to define it now.

---

➤ **To create an analog alarm status color link:**

1. Double-click the object or select it, and then on the **Special** menu, click **Animation Links**. The link selection dialog box will appear.
  - ☞ To quickly access the animation link selection dialog box, right-click the object, and then click **Animation Links**.
2. In the **Fill Color (Line Color or Text Color)** section, click **Analog Alarm**. The **Fill Color -> Analog Tagname Alarm Status** dialog box appears:



3. In the **Tagname** box, type the analog (integer or real) tagname whose alarm status you want associated with the object.
  - ☞ Right-click the **Tagname** box, to access the commands that you can apply to the selected text.
4. In the **Alarm Type** group, select type of alarm that you want to associate with the object. There are three mutually exclusive types of analog color links that you can use:

**Value Alarm** - You can select up to five different colors depending on the status of the value alarms defined for the tagname (example above).

**Deviation** - You can select up to three different colors depending on the status of the deviation alarms defined for the tagname (example above).

**ROC (Rate-of-Change)** - You can select two different colors depending on the status of the rate-of-change alarm defined for the tagname.

- In the **Colors** group, click each color box to open the color palette. Click the color in the color palette that you want to use for each color state.

---

**Note** You must use solid colors for line and text color links. If you select a dithered (mixed) color, WindowMaker, by default, will select the closest solid color. To avoid dithered colors, your video card must have at least 2MB and your color depth settings must be higher than 256 colors, such as 32K or 65K (sometimes called "high color".)

---

*For more information on the color palette, see "Applying Color Links."*

- Click **OK** to attach the link to the object and return to the animation links dialog box. You can now attach another link to the object if desired.

---

**Note** If the tagname you entered is not defined in your Tagname Dictionary you will be prompted to define it now.

---

## Creating Object Size Links

You use **Object Size** links to vary the height and/or width of an object according to the value of an analog (integer or real) tagname or analog expression. Size links provide the ability to control the direction in which the object enlarges in height and/or width by setting the "anchor" for the link. Both height and width links can be attached to the same object.

---

**Note** The height and width links are created the same way. This procedure describes the **Height** link.

---

➤ **To create a height (or width) link:**

- Double-click the object or select it, and then on the **Special** menu, click **Animation Links**. The link selection dialog box will appear.

*To quickly access the animation link selection dialog box, right-click the object, and then click **Animation Links**.*

- In the **Object Size** section, click **Height**. The **Object Height -> Analog Value** dialog box appears:

- In the **Expression** box, type an analog (integer or real) tagname or an expression that equates to an analog value.

*Up to 256 characters may be typed for your expression.*

Right-click the **Expression** box, to access the commands that you can apply to the selected text.

4. In the **Value at Max Height** box, type the value of the tagname or expression that will result in the object reaching its maximum height.
5. In the **Value at Min Height** box, type the value of the tagname or expression that will result in the object reaching its minimum height.
6. In the **Max % Height** box, type the percentage (0-100) of its height that the object will be when the tagname or expression reaches the value set in the **Value at Max Height** field.
7. In the **Min % Height** box, type the percentage (0-100) of its height that the object will be when the tagname or expression reaches the value set in the **Value at Min Height** field.
  - ☞ The percent height figures are expressed as a percentage of the actual "drawn size" of the object, which is 100%.
8. Select the **Anchor** point from which the object will enlarge in height.
  - ☞ Selecting **Top** will cause the object to be enlarged from its top downward. Selecting **Middle** will cause the object to be enlarged from its centerpoint outwards in both directions. Selecting **Bottom** will cause the object to be enlarged from its bottom upwards.
9. Click **OK** to attach the link to the object and return to the animation links dialog box. You can now attach another link to the object if desired.

---

**Note** If the tagname you entered is not defined in your Tagname Dictionary you will be prompted to define it now.

---

## Creating Location Links

You use **Location Links** to make an object automatically move horizontally, vertically, or in both directions in response to changes in the value of an analog tagname or expression.

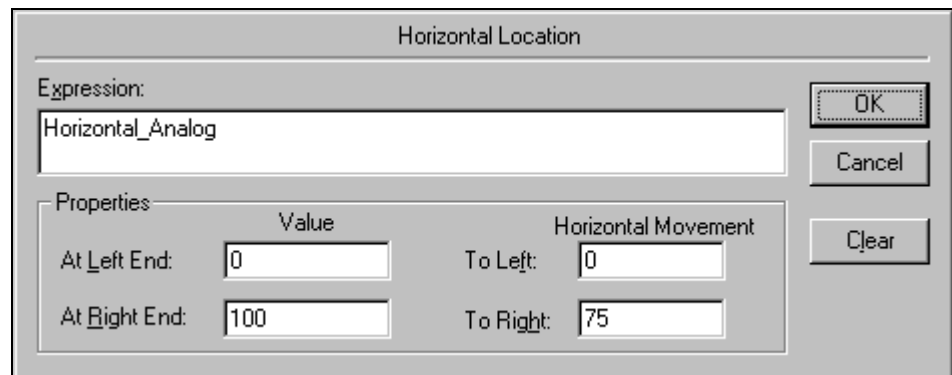
---

**Note** The **Horizontal** and **Vertical Location** links are created the same way. This procedure describes the **Horizontal Location** link.

---

➤ **To create a horizontal location link:**

1. Double-click the object or select it, and then on the **Special** menu, click **Animation Links**. The link selection dialog box will appear.
  - ☞ To quickly access the animation link selection dialog box, right-click the object, and then click **Animation Links**.
2. In the **Location** section, click **Horizontal**. The **Horizontal Location** dialog box appears:





3. In the **Expression** box, type an analog (integer or real) tagname or an expression that equates to an analog value.
  - ☞ Up to 256 characters may be typed for your expression.  
Right-click the **Expression** box, to access the commands that you can apply to the selected text.
4. In the **At Left End** box, type the value for the tagname when the object is located at its farthest left position.
5. In the **At Right End** box, type the value for the tagname when the object is located at its farthest right position.
6. In the **To Left** box, type the number of pixels the object can move to the left of its drawn position.
  - ☞ At the far left position, the tagname's value will be equal to the value entered in the **At Left End** field.
7. In the **To Right** box, type the number of pixels the object can move to the right of its drawn position.
  - ☞ At the far right position, the tagname's value will be equal to the value entered in the **At Right End** field.
8. Click **OK** to attach the link to the object and return to the animation links dialog box. You can now attach another link to the object if desired.

---

**Note** If the tagname you entered is not defined in your Tagname Dictionary you will be prompted to define it now.

---

## Creating Percent Fill Links

You use **Percent Fill Links** to provide the ability to vary the fill level of a filled shape (or a symbol containing filled shapes) according to the value of an analog tagname or an expression that computes to an analog value. For example, this link may be used to show the level of liquids in a vessel. An object or symbol may have a horizontal fill link, a vertical fill link, or both.

---

**Note** The **Horizontal** and **Vertical Percent Fill** links are created the same way. This procedure describes the **Vertical Percent Fill** link.

---

- **To create a vertical percent fill link:**
1. Double-click the object or select it, and then on the **Special** menu, click **Animation Links**. The link selection dialog box will appear.
    - ☞ To quickly access the animation link selection dialog box, right-click the object, and then click **Animation Links**.
  2. In the **Percent Fill** section, click **Vertical**. The **Vertical Fill -> Analog Value** dialog box appears:

3. In the **Expression** box, type an analog (integer or real) tagname or an expression that equates to an analog value.
  - ☞ Up to 256 characters may be typed for your expression.
    - Right-click the **Expression** box, to access the commands that you can apply to the selected text.
4. In the **Value at Max Fill** type the value the expression that will result in the object being filled to its maximum level.
5. In the **Value at Min Fill** type the value the expression that will result in the object being filled to its minimum level.
6. In the **Max % Fill** box, type the percentage (0-100) that the object will be filled when the expression reaches the level set in the **Value at Max Fill** box.
  - ☞ If the value of the expression is greater than this number, it will be ignored.
7. In the **Min % Fill** box, type the percentage (0-100) that the object will be filled when the expression reaches the level set in the **Value at Min Fill** box.
  - ☞ If the value of the expression is greater than this number, it will be ignored.
8. Select the **Direction** that you want the object to fill from.
  - ☞ If **Up** is selected, it will be filled from the bottom to the top. If **Down** is selected, it will be filled from the top to the bottom.
9. In the **Background Color** box to open the color palette. Click on the desired color. The color palette will be removed from the screen.
  - ☞ This **Background Color** selection is for the color of the **unfilled** portion of the object. The actual fill color is the color that you select for the object when you draw it. If you link both **Vertical Percent Fill** and **Horizontal Percent Fill** links, to the same object, the last color you select in either of their link dialog boxes will be used as the background color.
10. Click **OK** to attach the link to the object and return to the animation links dialog box. You can now attach another link to the object if desired.

---

**Note** If the tagname you entered is not defined in your Tagname Dictionary you will be prompted to define it now.

---

## Creating Miscellaneous Links

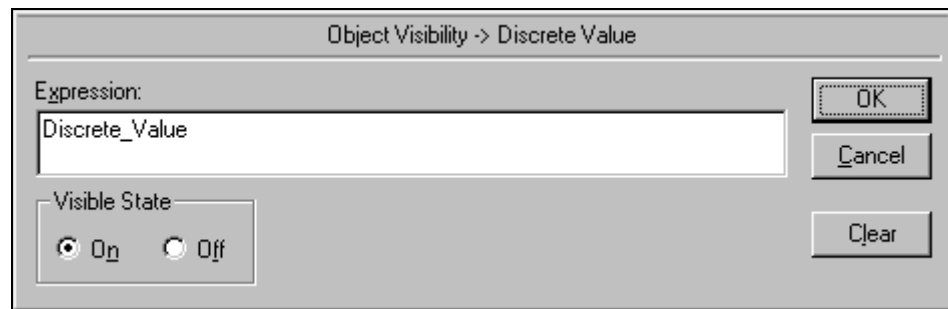
There are four type of miscellaneous links.

Misc Link	Description
-----------	-------------

<b>Visibility</b>	Use to control the visibility of an object based on the value of a discrete tagname or expression.
<b>Blink</b>	Used to make an object blink based on the value of a discrete tagname or expression.
<b>Orientation</b>	Used to make an object rotate based on the value of a tagname or expression.
<b>Disable</b>	Used to disable the touch functionality of objects based on the value of a tagname or expression. ☞ Often used as part of a security strategy.

➤ **To create a visibility link**

1. Double-click the object or select it, and then on the **Special** menu, click **Animation Links**. The link selection dialog box will appear.
  - ☞ To quickly access the animation link selection dialog box, right-click the object, and then click **Animation Links**.
2. In the **Miscellaneous** section, click **Visibility**. The **Object Visibility -> Discrete Value** dialog box appears:



3. In the **Expression** box, type a discrete tagname or an expression that equates to a discrete value.
  - ☞ Up to 256 characters may be typed for your expression.

Discrete expressions can also contain analog tagnames, for example, TankLevel >= 75. In this example, when the value of the tagname, TankLevel is greater than or equal to 75, the object will become visible in the window.

Right-click the **Expression** box, to access the commands that you can apply to the selected text.
4. Select the **Visible State** for the object. If you select **On**, the object will be invisible when the value of the expression is true. If you select **Off**, the object will be visible when the value of the expression is true.
5. Click **OK** to attach the link to the object and return to the animation links dialog box. You can now attach another link to the object if desired.

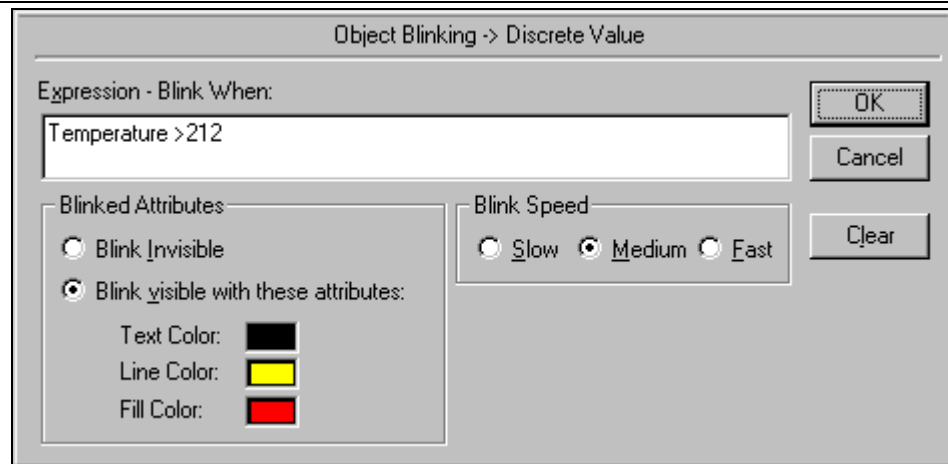
---

**Note** If the tagname you entered is not defined in your Tagname Dictionary you will be prompted to define it now.

---

➤ **To create a blink link:**

1. Double-click the object or select it, and then on the **Special** menu, click **Animation Links**. The link selection dialog box will appear.
  - ☞ To quickly access the animation link selection dialog box, right-click the object, and then click **Animation Links**.
2. In the **Miscellaneous** section, click **Blink**. The **Object Blinking -> Discrete Value** dialog box appears:



3. In the **Expression - Blink When** box, type a discrete tagname or an expression that equates to a discrete value.

☞ Up to 256 characters may be typed for your expression.

Discrete expressions can also contain analog tagnames. For example, TankLevel >= 75. In this example, when the value of the variable "TankLevel" is greater than or equal to "75," the object will blink.

Right-click the **Expression** box, to access the commands that you can apply to the selected text.

4. Select the **Blinked Attributes** that you want for the object.

If you select **Blink Invisible**, the object/symbol blinks by disappearing and reappearing in the window. If you select **Blink visible with these attributes**, the object/symbol remains visible in the window and the changing of the color attributes selected creates the blinking effect.

Click the **Text Color**, **Line Color** and **Fill Color** boxes to open the color palette. Click on the desired color. The color palette will be removed from the screen.

☞ Choosing a "fill" blink color that is the same as the object's "fill" color will not allow the object to "blink."

5. Select the **Blink Speed** that you want to use for the blinking speed of the object.

---

**Note** To configure the blink speed for **Slow**, **Medium**, and **Fast**, on the **Special** menu, point to **Configure** and then click **Window Viewer**. The **Window Viewer General** property sheet will appear. In the **Blink Frequency** group, type the number of milliseconds you want to use for the speeds.

Any changes you make to these settings are global and will effect the blink speeds of all blink links throughout your application.

---

6. Click **OK** to attach the link to the object and return to the animation links dialog box. You can now attach another link to the object if desired.

---

**Note** If the tagname you entered is not defined in your Tagname Dictionary you will be prompted to define it now.

---

#### ➤ To create an orientation link

1. Double-click the object or select it, and then on the **Special** menu, click **Animation Links**. The link selection dialog box will appear.

☞ To quickly access the animation link selection dialog box, right-click the object, and then click **Animation Links**.

- In the **Miscellaneous** section, click **Orientation**. The **Orientation -> Analog Value** dialog box appears:

- In the **Expression** box, type an analog (integer or real) tagname or an expression that equates to an analog value.
    - Up to 256 characters may be typed for your expression.
      - Right-click the **Expression** box, to access the commands that you can apply to the selected text.
  - In the **Value at Max CCW** box, type the value the expression must be for the object to be rotated to its maximum counter-clockwise position.
    - If the value of the expression is greater than this number, it will be ignored.
  - In the **Value at Max CW** box, type the value the expression must be for the object to be rotated to its maximum clockwise position.
    - If the value of the expression is greater than this number, it will be ignored.
  - In the **CCW Rotation** box, type the degrees the object will rotate counter-clockwise when the **Value at Max CCW** is reached.
  - In the **CW Rotation** box, type the degrees the object will rotate clockwise when the **Value at Max CW** is reached.
    - The object is rotated clockwise or counter-clockwise based on its original drawn position when drawn in WindowMaker.
      - To force an object like text to a specific angle, simply set **Value at Max CCW** to 360 and **Value at Max CW** to 0, **CCW Rotation** to 360 and **CW Rotation** to 0, and then in the **Expression** box, type the angle value such as 90 (for 90 degrees). Remember, without a tagname, this expression will never change and the object will always hold its 90 degree position.
- 
- Note** Text can be set in WindowMaker, but not rotated in WindowViewer on a tagname value.
- 
- In the **X Position** box, type the number of pixels the rotation centerpoint is to be moved horizontally from the centerpoint of the object. (Positive values are to the right of centerpoint.)
    - The orientation link uses the center of the object or symbol as the center of rotation.
  - In the **Y Position** box, type the number of pixels the rotation centerpoint is to be moved vertically from the centerpoint of the object. (Positive values are to the left of centerpoint.)

- Click **OK** to attach the link to the object and return to the animation links dialog box. You can now attach another link to the object if desired.

**Note** If the tagname you entered is not defined in your Tagname Dictionary you will be prompted to define it now.

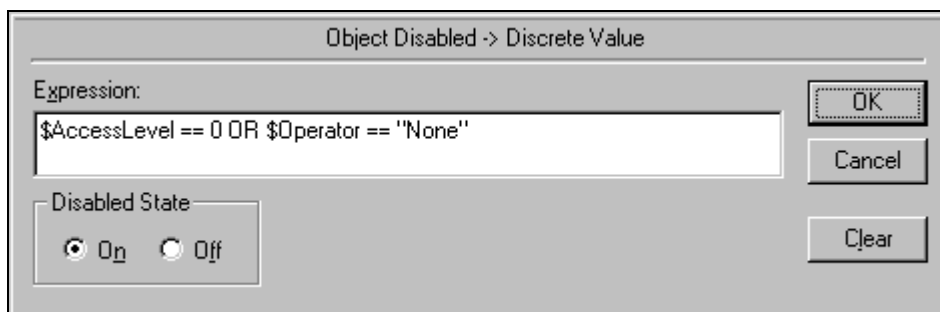
➤ **To create a disable link:**

☞ The disable link is very useful when you are applying security to your application. For example, you can disable objects based upon the logged on operator's access level or name.

- Double-click the object or select it, and then on the **Special** menu, click **Animation Links**. The link selection dialog box will appear.

☞ To quickly access the animation link selection dialog box, right-click the object, and then click **Animation Links**.

- In the **Miscellaneous** section, click **Disable**. The **Object Disabled -> Discrete Value** dialog box appears:



- In the **Expression** box, type a discrete tagname or an expression that equates to a discrete value.

☞ By using the above expression if no one is logged on, the object or button is secured from tampering.

Up to 256 characters may be typed for your expression.

Discrete expressions can also contain analog tagnames. For example, TankLevel >= 75. In this example, when the value of the variable "TankLevel" is greater than or equal to "75," the object will be disabled.

Right-click the **Expression** box, to access the commands that you can apply to the selected text.

- Select the **Disabled State** that will turn off or on functionality of the object when the discrete tagname or expression is true.

☞ A disabled state of "on" means the touch functionality of the object or button are turned off and cannot be clicked as long as the expression is true.

- Click **OK** to attach the link to the object and return to the animation links dialog box. You can now attach another link to the object if desired.

**Note** If the tagname you entered is not defined in your Tagname Dictionary you will be prompted to define it now.

## Creating Value Display Links

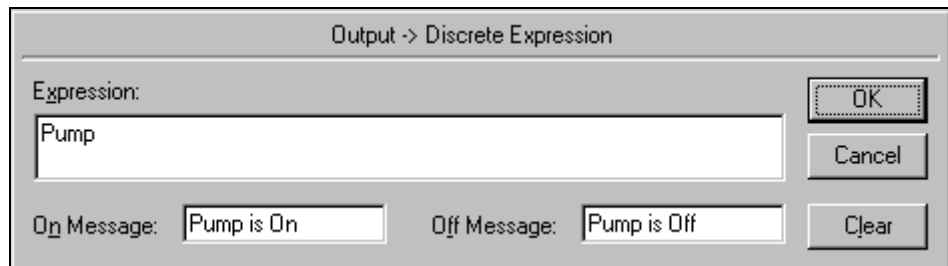
Value Display Links provide the ability to use a text object to display the value of a discrete, analog, or string tagname. There are three types:

Value Display Type	Description
--------------------	-------------

- Discrete** Uses the value of a discrete expression to display an On or Off user defined message in a text object.
- Analog** Displays the value of an analog expression in a text object.
- String** Displays the value of a string expression in a text object.

➤ **To create a discrete value display link:**

1. Double-click the object or select it, and then on the **Special** menu, click **Animation Links**. The link selection dialog box will appear..
  - ☞ To quickly access the animation link selection dialog box, right-click the object, and then click **Animation Links**.
2. In the **Value Display** section, click **Discrete**. The **Output -> Discrete Expression** dialog box appears:



3. In the **Expression** box, type a discrete tagname or an expression that equates to a discrete value.
  - ☞ Discrete expressions can also contain analog tagnames. For example, TankLevel >= 75. In this example, when the value of the variable "TankLevel" is greater than or equal to "75," the appropriate message will be displayed.  
Up to 256 characters may be typed for your expression.  
Right-click the **Expression** box, to access the commands that you can apply to the selected text.
4. In the **On Message** box, type the message that you want displayed when the value of the discrete expression equals 1 (True, On, Yes).
5. In the **Off Message** box, type the message that you want displayed when the value of the discrete expression equals 0 (False, Off, No).
  - ☞ The messages will be displayed in the location of the original text object using the font, size, color, alignment and linked attributes set for that object. The original contents of the field have no effect on the displayed message at runtime.
6. Click **OK** to attach the link to the object and return to the animation links dialog box. You can now attach another link to the object if desired.

---

**Note** If the tagname you entered is not defined in your Tagname Dictionary you will be prompted to define it now.

---

- ☞ You can also use a **Value Display Output -> String Expression** link to display the on and off messages for a discrete tagname. For the link, you would type the following expression:

```
DText (Pump, Pump.OnMsg, Pump.OffMsg);
```

In this expression, the **.OnMsg** and **.OffMsg** strings will be extracted from the OpenHMI Tagname Dictionary definition for the discrete tagname, Pump.

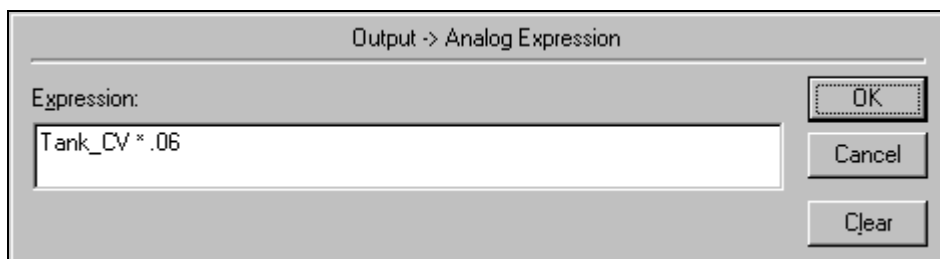
➤ **To create an analog value display link:**

☞ For more information on formatting analog display objects, see Chapter 2 - Using WindowMaker.

1. Double-click the object or select it, and then on the **Special** menu, click **Animation Links**. The link selection dialog box will appear.

☞ To quickly access the animation link selection dialog box, right-click the object, and then click **Animation Links**.

2. In the **Value Display** section, click **Analog**. The **Output -> Analog Expression** dialog box appears:



3. In the **Expression** box, type an analog (integer or real) tagname or an expression that equates to an analog value. (You can also use a discrete type tagname in this expression. It will simply display a 1 or 0.)

☞ Up to 256 characters may be typed for your expression.

Right-click the **Expression** box, to access the commands that you can apply to the selected text.

4. Click **OK** to attach the link to the object and return to the animation links dialog box. You can now attach another link to the object if desired.

---

**Note** If the tagname you entered is not defined in your Tagname Dictionary you will be prompted to define it now.

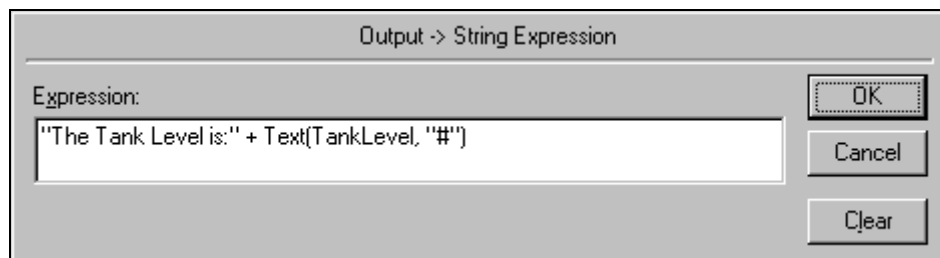
---

➤ **To create a string value display link:**

1. Double-click the object or select it, and then on the **Special** menu, click **Animation Links**. The link selection dialog box will appear.

☞ To quickly access the animation link selection dialog box, right-click the object, and then click **Animation Links**.

2. In the **Value Display** section, click **String**. The **Output -> String Expression** dialog box appears:



3. In the **Expression** box, type a message tagname or an expression that equates to a message tagname.

☞ In the above expression, the function **Text()** is used to convert the value of the integer tagname, TankLevel, to a string.

In Up to 256 characters may be typed for your expression.



---

Right-click the **Expression** box, to access the commands that you can apply to the selected text.

4. Click **OK** to attach the link to the object and return to the animation links dialog box. You can now attach another link to the object if desired.

---

**Note** If the tagname you entered is not defined in your Tagname Dictionary you will be prompted to define it now.

---

## CHAPTER 5

# Creating QuickScripts in OpenHMI

OpenHMI scripting is one of the most powerful features of an OpenHMI application. The OpenHMI QuickScript capabilities allow you to execute commands and logical operations based on specified criteria being met. For example, a key being pressed, a window being opened, a value changing, and so on.

By using scripts, a wide variety of customized and automated system functions can be created.

# OpenHMI QuickScripts

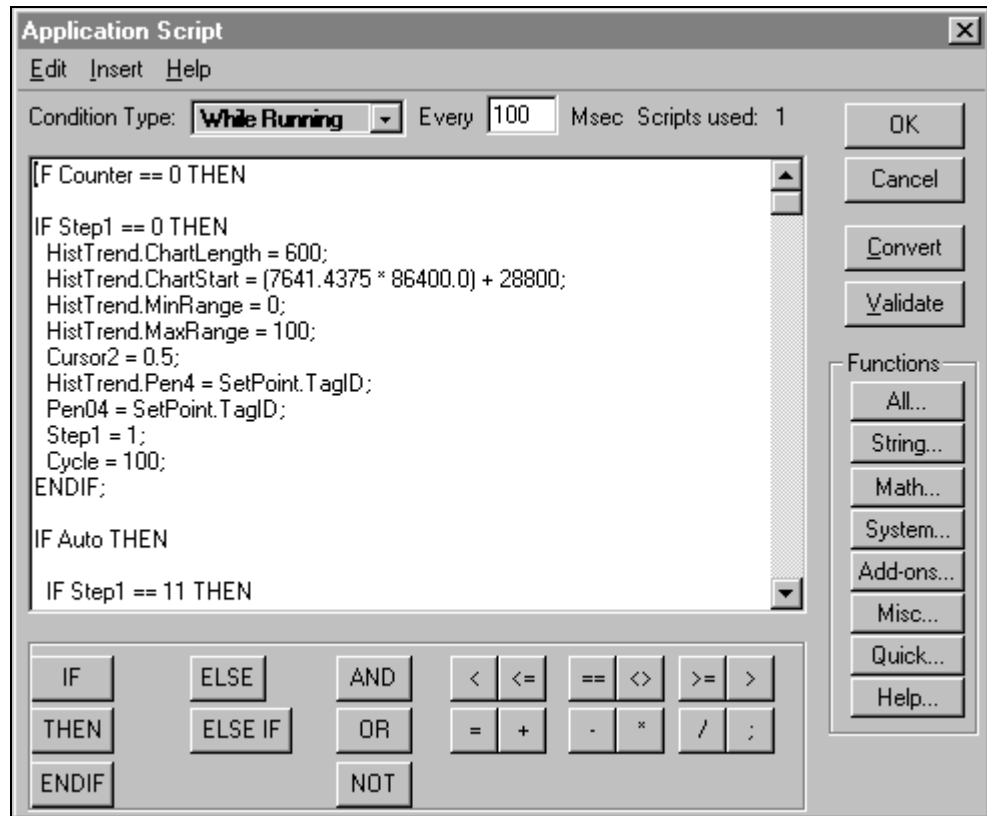
All OpenHMI QuickScripts are event driven. The event may be a data change, condition, mouse click, timer, and so on. The order of processing is application specific. While it may appear that there is some inherent order in the way multiple scripts initiated by the same event are scheduled, there is no guarantee of any specific order. Therefore, you should not build any dependency on the order of processing.

The following briefly describes the types of scripts that you can create:

<b>Script Type</b>	<b>Description</b>
<b>Application</b>	Linked to the entire application.
<b>Window</b>	Linked to a specific window.
<b>Key</b>	Linked to a specific key or key combination on the keyboard.
<b>Condition</b>	Linked to a discrete tagname or expression.
<b>Data Change</b>	Linked to a tagname and/or <b>tagname.field</b> only.
<b>Action Pushbutton</b>	Associated with an object that you link to an <b>Touch Link - Action Pushbutton</b> .
<b>ActiveX Event</b>	Execute ActiveX control events in runtime.

## Using the OpenHMI QuickScript Editor

The OpenHMI QuickScript editor is basically the same for all script types. Therefore, to avoid redundancy, its common functions and features are described in this section. The items that are unique to a script type are described in that script type's respective section later in this chapter.



## QuickScript Editor Common Procedures

This section describes the generic procedures that you will use when writing scripts in the various OpenHMI QuickScript editor dialog boxes. The procedures that are unique to a script type are described in that script type's respective section later in this chapter.

There are text, equivalency and mathematical operator buttons at the bottom of the QuickScript editor that you can click to quickly insert the displayed keyword, function or symbol into your script at the cursor location.

➤ **To indent/unindent text in a script:**

Position the cursor at the beginning of the line that you want to indent, and then press the TAB key. To remove the indent, press hold down the SHIFT key while you press the TAB key.

➤ **To create a new script:**

On the **Script** menu, click **New**.

---

**Note** The **Script** menu does not exist for **Application**, **Window Scripts** or **Touch Pushbutton Action** scripts.

---

➤ **To delete a script from your application:**

Select the text you want to delete, and then on the **Script** menu, click **Erase**. The script is deleted from your application entirely.

---

**Note** Deleted text is not written to the Windows Clipboard.

---

➤ **To undo your last action:**

On the **Edit** menu, click **Undo**. Your last editing operation, a paste, for example, is reversed.

☞ To quickly execute this command, right-click the script window, and then click **Undo**. The **Undo** command will not be active unless you have performed an action that can be reversed.

➤ **To select the entire script:**

On the **Edit** menu, click **Select All**. The entire script is selected.

☞ To quickly execute the command, right-click the script window, and then click **Select All**. You can now copy, cut or delete the entire script.

➤ **To cut selected text from a script:**

Select the text you want to remove, and then on the **Edit** menu, click **Cut**. The cut text is deleted from the script and copied to the Windows Clipboard. You can now paste the cut text at another location in this script or you can paste it in another script.

☞ To quickly perform this command, right-click the script window, and then click **Cut**. The **Cut** command will not be active unless you have selected text to cut.

➤ **To copy selected text from a script:**

Select the text to be removed, and then on the **Edit** menu, click **Copy**. The copied text is written to the Windows Clipboard. You can now paste the copied text at another location in this script, or you can paste it in another script.

☞ To quickly perform this command, right-click the script window, and then click **Copy**. The **Copy** command will not be active unless you have selected text to copy.

---

**Note** When you cut or copy text, it is automatically written to the Windows Clipboard. This information remains on the Clipboard until you perform a subsequent cut or copy command.

---

➤ **To paste text into a script:**

On the **Edit** menu, click **Paste**. The contents of the Windows Clipboard is pasted into your script at the cursor location.

☞ To quickly execute the **Paste** command, right-click the script window, and then click **Paste**. (The **Paste** command will not be active if there is nothing in the Windows Clipboard to paste.)

➤ **To clear the text in a script:**

On the **Edit** menu, click **Clear**. All the text in the script is erased. However, the script is not deleted from your application. If you select this command then close the script editor and reopen it, the script will reappear.

☞ To completely delete the script, you must use the **Erase** command on the **Script** menu or select the entire script, then right-click a blank area of the script window, and then click **Delete**.

➤ **To insert a function into a script:**

1. On the **Insert** menu, point to **Functions**, and then click the name of the function category. The respective **Choose function** dialog box will appear.

2. Click the function that you want to use. The dialog box will close and the function will automatically be inserted into your script at the cursor location.

The types of functions available are:

Function	Description
<b>All</b>	<p>The <b>Choose function</b> dialog box appears displaying all available functions including the functions for each installed add-on program (Recipe Manager, SPC Pro and SQL Access Manager).</p> <p>☞ You can also click the <b>All</b> button in the <b>Functions</b> group to access these functions.</p>
<b>String</b>	<p>The <b>Choose function</b> dialog box appears displaying all available string functions.</p> <p>☞ You can also click the <b>String</b> button in the <b>Functions</b> group to access these functions.</p>
<b>Math</b>	<p>The <b>Choose function</b> dialog box appears displaying all available mathematical functions.</p> <p>☞ You can also click the <b>Math</b> button in the <b>Functions</b> group to access these functions.</p>
<b>System</b>	<p>The <b>Choose function</b> dialog box appears displaying all available system functions. For example, the functions to start and/or activate another application, read and/or write file and disk information, and so on.</p> <p>☞ You can also click the <b>System</b> button in the <b>Functions</b> group to access these functions.</p>
<b>Add-ons</b>	<p>The <b>Choose function</b> dialog box appears displaying all available functions for each installed add-on program (Recipe Manager, SPC Pro and SQL Access Manager).</p> <p>☞ You can also click the <b>Add-ons</b> button in the <b>Functions</b> group to access these functions.</p>
<b>Misc</b>	<p>The <b>Choose function</b> dialog box appears displaying all available miscellaneous functions. For example, the functions for alarms, windows controls, ActiveX controls, and so on.</p> <p>☞ You can also click the <b>Misc</b> button in the <b>Functions</b> group to access these functions.</p>
<b>Help</b>	<p>The <b>Choose function to Obtain Help for</b> dialog box appears listing all available functions. Click a function to open its respective Help topic.</p> <p>☞ You can also click the <b>Help</b> button in the <b>Functions</b> group to access these functions.</p>

☞ For more information on the individual script functions, see "Script Functions."

➤ **To insert a tagname into a script:**

1. On the **Insert** menu, click **Tagname**. The Tag Browser will appear in the unlimited selection mode.

---

**Note** The tagnames defined in the last tag source accessed through the Tag Browser will be displayed. To change the tag source, click the **Tag Source** arrow and select a different tag source in the list.

Click the Define Tag Sources button to add or remove a tag source from the **Tag Source** list.

---

2. Double-click the tagname you want to use or select it, and then click **OK**. The Tag Browser will close and the tagname will automatically be inserted into your QuickScript at the cursor location.

☞ To quickly access the Tag Browser, double-click a blank area in the QuickScript window.

To access a specific tagname's definition in the Tagname Dictionary, type the tagname in the QuickScript window, and then double-click it.

☞ For more information on the Tag Browser, see Chapter 3 - Tagname Dictionary.

➤ **To insert a tagname .field into a script:**

1. On the **Insert** menu, click **Tagname**. The Tag Browser will appear in the unlimited selection mode.

---

**Note** The tagnames defined in the last tag source accessed through the Tag Browser will be displayed. To change the tag source, click the **Tag Source** arrow and select a different tag source in the list.

Click the Define Tag Sources button to add or remove a tag source from the **Tag Source** list.

---

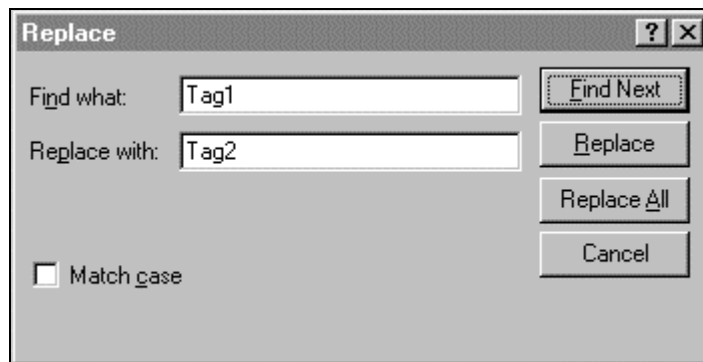
2. Select the tagname that you want to use, and then click the **Dot Field** arrow. Select the **.field** that you want to use with the tagname in the list.
3. Click **OK**. The selected tagname.**field** will be inserted into your QuickScript at the cursor location.

☞ To quickly insert a tagname **.field**, type the tagname followed by a period (.), and then double-click to the right of the period. The **Choose field name** dialog box will appear. Click the **.field** that you want to use. The dialog box will close and the selected **.field** will automatically be inserted into your QuickScript at the cursor location.

☞ For more information on the Tag Browser, see Chapter 3 - Tagname Dictionary.

➤ **To find or replace a tagname in a script:**

4. On the **Edit** menu, click **Find**. The **Replace** dialog box appears:



5. In the **Find what** box, type the tagname that you want to find (or replace), and then click **Find Next**.
6. In the **Replace with** box, type the new tagname that you want to use to replace the old tagname then click **Replace** or **Replace All**.

☞ If you only want to replace certain instances of an old tagname, click **Find Next**. **OpenHMI** will begin searching your script for the old tagname. When the old tagname is found, it will be highlighted. Click **Replace** to replace it with the new tagname or click **Find Next** to skip it and continue searching. If

you want to replace all occurrences of a specific tagname, click **Replace All** at any time during the search.

7. Select the **Match case** option if you find specific upper or lowercase instances of the tagname.
8. Click **Cancel** to close the dialog box.

➤ **To insert a window name into a script:**

1. On the **Insert** menu, click **Window**. The **Window Name to Insert** dialog box will appear displaying the names of all the windows in your application.
2. Click the window name that you want to use. The dialog box will close and the window name will automatically be inserted into your script at the cursor location.

➤ **To validate a script:**

Click **Validate** to verify that your script syntax is accurate at any time while you are writing the script.

- ☞ Validation is automatically performed when you click **OK** or **Save**. If the system encounters errors when validating your script, a corresponding error message box will appear.

☞ For more information on script errors, see "Script Editor Error Messages."

➤ **To save a script:**

If you are writing multiple scripts, after you have finished writing one, you can click **Save** to save it, and then on the **Script** menu, click **New** to write another new script.

---

**Note Application** and **Window** scripts don't support this function. Otherwise, the save function is automatically performed when you click the **OK** button.

---

- ☞ If the system encounters errors when saving your script, a corresponding error message box will appear.

➤ **To restore a script:**

If you change a script and decide that you want to clear your changes and restore the original script, click **Restore**.

---

**Note** You cannot restore a script once you have saved it. **Application** and **Window** scripts don't support this function.

---

➤ **To exit the script editor:**

On the **Script** menu, click **Exit**. The script editor will close and the script will be saved unless an error is encountered.

- ☞ You can also close the script editor by clicking **OK** once you have completed writing your script.

---

**Note** When you select **Exit**, **OK** or you click the **X** button in the upper right hand corner of the dialog box, the system automatically verifies your script for accuracy.

---

☞ For more information on script errors, see "Script Editor Error Messages."

➤ **To specify a script's execution frequency:**

In the **While Running/Showing/Down Every 0 Milliseconds** boxes, type the number of milliseconds that you want to elapse before the script executes.

- ☞ When you create an Application **While Running** script, Window **While Showing** scripts, Condition **While On True/On False** scripts or Key and Touch Pushbutton Action **While Down** scripts you must specify the frequency (in milliseconds) that they will be executed.



---

**Note** WindowViewer will make every attempt possible to run these types of scripts as fast as the time you specify. However, the performance cannot be guaranteed. Also scripts can never run any faster than the **Tick Interval** setting that you specify when you configure WindowViewer's properties.

Scripts cannot execute faster than every 10 milliseconds on the Windows NT operating system or every 50 milliseconds on Windows 95.

---

*↪* For more information on the **Tick Interval** setting, see Chapter 2 - Using WindowMaker.

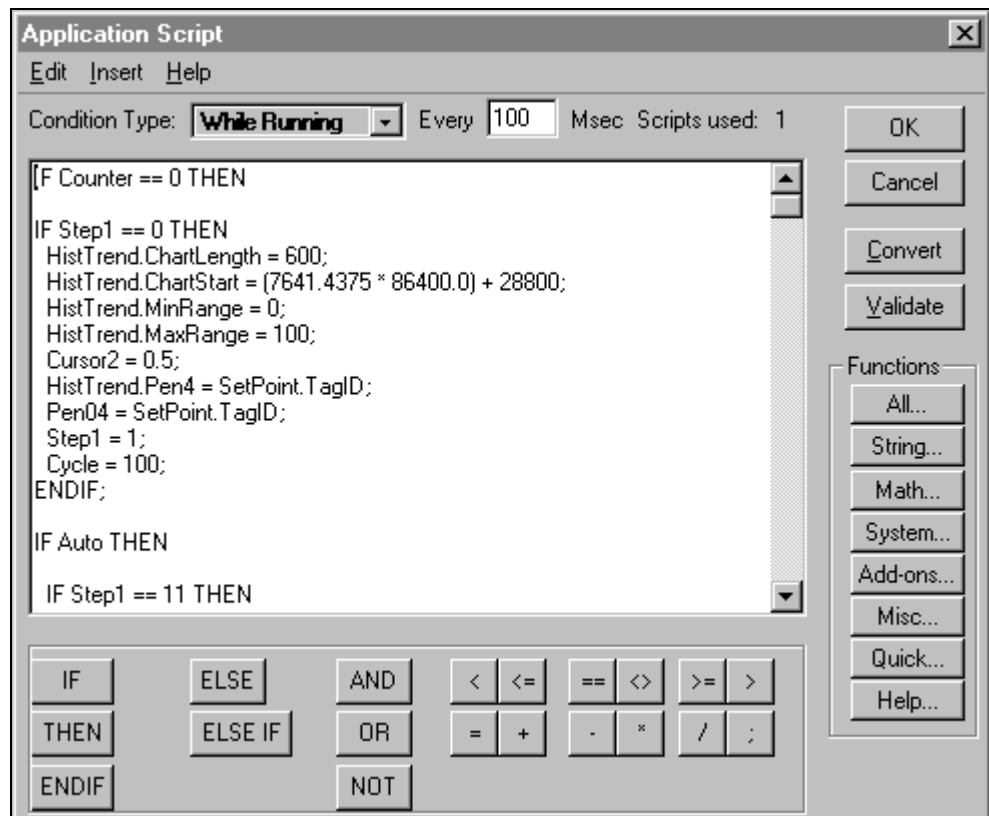
# Application Scripts

The Application Scripts are linked to the entire application. You can use application scripts to start other applications, create process simulations, calculate variables, and so on. There are three types of Application Scripts that you can apply to an application:

- On Startup**            Executes one time when the application is initially started up.
- While Running**       Executes continuously at the specified frequency while the application is running.
- On Shutdown**        Executes one time when the application is exited.

**Note** The Application **On Startup** script executes before any window opens or any runtime initialization occurs. Therefore, you cannot refer to ActiveX methods, properties or events in an Application **On Startup** script. Similarly, I/O communications are initialized after the Application **On Startup** script executes. Therefore, you cannot refer to I/O type tagnames in an Application **On Startup** script. In addition, I/O type tagnames will not update in an Application **On Shutdown** script. Also, you cannot use an Application **On Shutdown** script to startup other applications.

- **To access the Application Script editor:**  
 On the **Special** menu, point to **Scripts**, and then click **Application Scripts**, or in the Application Explorer under **Scripts**, double-click **Application**. The **Application Script** editor appears:
  - ☞ In the Application Explorer under **Scripts**, you can also right-click **Application**, and then click **Open**.



When you select a **While Running** script, the **Every 0 Milliseconds** box becomes active. In the box, type the number of milliseconds that you want to elapse before the script executes. If you want the script to execute immediately, create an identical **On Startup** script. However, as long as the condition or event for the **While Running** script is met, the script will repeatedly execute at the specified frequency.

# Window Scripts

Window Scripts are linked to a specific window. There are three types of scripts that you can apply to a window:

- On Show**                      Executes one time when the window is initially shown.
- While Showing**            Executes continuously at the specified frequency while the window is showing.
- On Hide**                        Executes one time when the window is hidden.

---

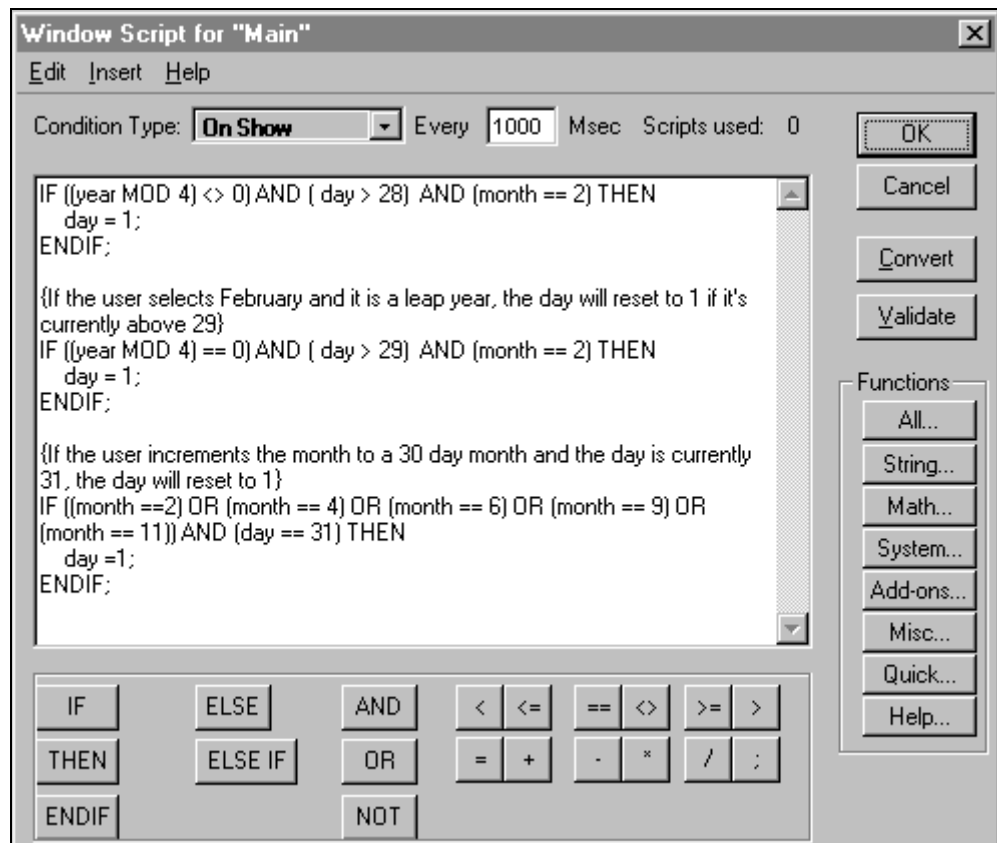
**Note** If you attach a Window Script to the active window and then you create a new window, the scripts from the active window can be copied to the new window. A message dialog box will appear asking if you want to copy the script(s).

---

➤ **To create Window Scripts:**

On the **Special** menu, point to **Scripts**, and then click **Window Scripts**. The **Window Script** editor appears:

- ☞ To quickly access the Window Script editor for a specific window, in the Application Explorer, under **Windows**, right-click the window name, and then click **Window Scripts**. You can also right-click a blank area of an open window, and then click **Window Scripts**. If a script exists for the selected window, it will be displayed.



When you select **While Showing**, the **Every 0 Milliseconds** box becomes active. In the box, type the number of milliseconds that you want to elapse before the script executes. If you want the script to execute immediately, create an identical **On Show** script. However, as long as the condition or event for the **While Showing** script is met, the script will repeatedly execute at the specified frequency.

# Key Scripts

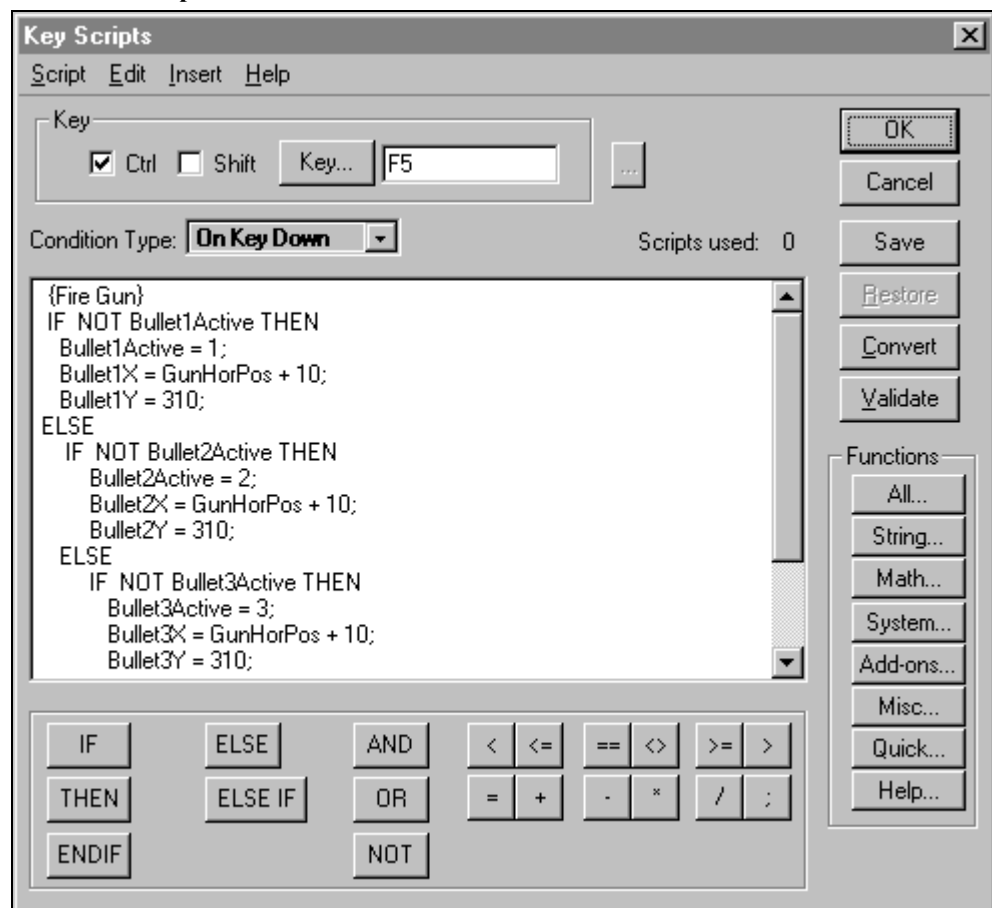
Key Scripts are linked to a specific key or key combination on the keyboard. You can use them to create global keys for the application. For example, returning to a main menu window, logging off the operator, and so on. There are three types of Key Scripts that you can apply to a key:

- On Key Down**      Executes one time when the key is initially pushed down.
- While Down**      Executes continuously at the specified frequency while the key is held down.
- On Key Up**        Executes one time when the key is released.

➤ **To access the Key Script editor:**

On the **Special** menu, point to **Scripts**, and then click **Key Scripts**, or in the Application Explorer under **Scripts**, double-click **Key**. The **Key Script** editor appears:

☞ In the Application Explorer, under **Scripts**, you can also right-click **Key**, and then click **Open**.



When you select **While Down**, the **Every 0 Milliseconds** box becomes active. In the box, type the number of milliseconds that you want to elapse before the script executes. If you want the script to execute immediately, create an identical **On Key Down** script. However, as long as the condition or event for the **While Down** script is met, the script will repeatedly execute at the specified frequency.

☞ For more information on assigning a key to the script, see "Assigning a Key Equivalent to a Script."

**Note** The key equivalents used in the local active window's for Touch Pushbutton Action scripts will override any global Key Scripts with the same key equivalents.

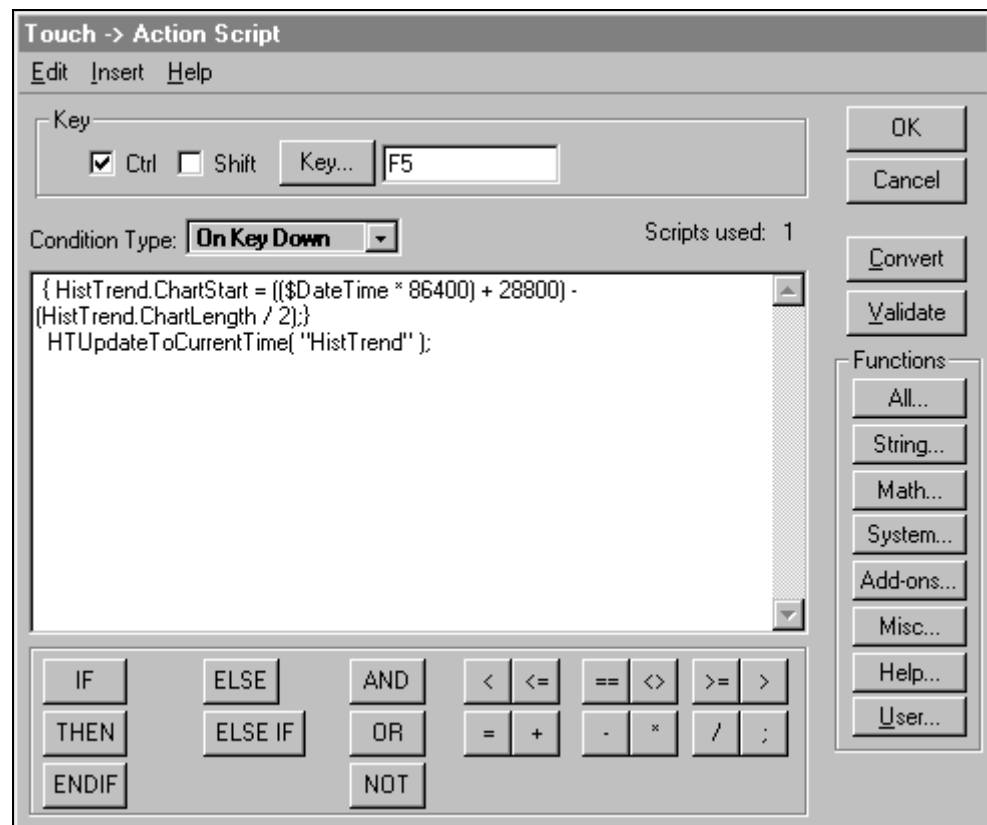
## Touch Pushbutton Action Scripts

Touch Pushbutton Action Scripts are similar to Key Scripts, except they are associated with an object that you link to a **Touch Link- Action Pushbutton**. (The script editor is accessed through the animation link selection dialog box.) They are executed when the operator clicks or presses the object or button assigned to the link. There are three types of Touch Action Scripts that you can apply to an object:

- On Key Down**      Executes one time when the key is initially pushed down.
- While Down**      Executes continuously at the specified frequency while the key is held down.
- On Key Up**        Executes one time when the key is released.

➤ **To create an action pushbutton script:**

1. Draw the object or button that you want to be linked to the script.
2. Double-click the object or select it, and then on the **Special** menu, click **Animation Links**. The animation link selection dialog box will appear.
  - ☞ To quickly access the dialog box, right-click the object, and then click **Animation Links**.
3. In the **Touch Pushbutton** section, click **Action**. The **OpenHMI -> Action Script** editor appears:



When you select **While Down**, the **Every 0 Milliseconds** box becomes active. In the box, type the number of milliseconds that you want to elapse before the script executes. If you want the script to execute immediately, create an identical **On Key Down** script. However, as long as the condition or event for the **While Down** script is met, the script will repeatedly execute at the specified frequency.

☞ For more information on assigning a key to the script, see "Assigning a Key Equivalent to a Script."

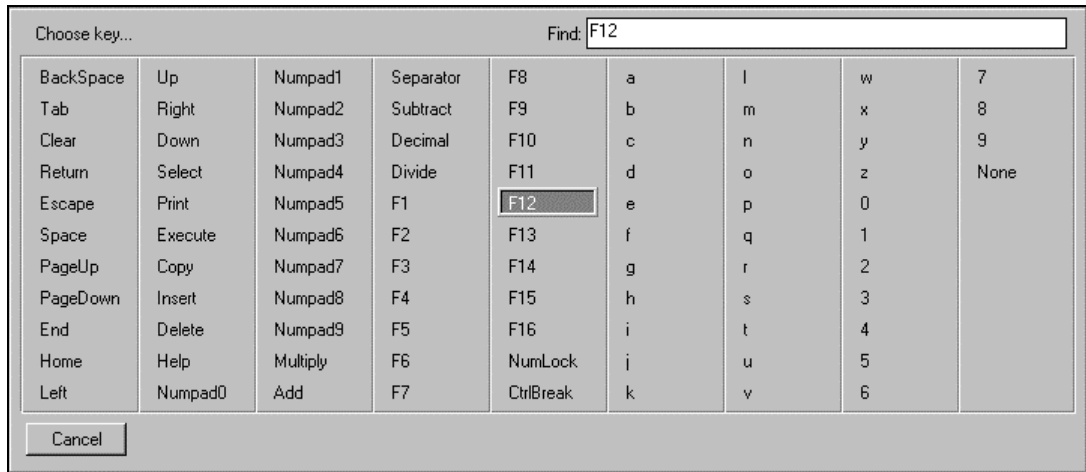
**Note** The key equivalents used in the local active window's for Touch Pushbutton Action scripts will override any global Key Scripts with the same key equivalents. Also, key equivalents are only active when the window with the object is active.

## Assigning a Key Equivalent to a Script

The Key Script and the Touch Action Script editors are a little different from the other QuickScript editors. Since you are creating scripts that apply to a key, you must specify the key(s) that you want the operator to press to execute the script.

➤ **To assign a key to a Key Script:**

1. Select **Ctrl** and/or **Shift**, if you want the operator to have to hold down the CTRL and/or SHIFT keys while pressing the key to execute the script.
2. Click **Key** to select the key you want assigned to the script. The **Choose key** dialog box appears:



3. Click the desired key. The dialog box automatically closes and your selection is automatically entered in the **Key** box.

# Condition Scripts

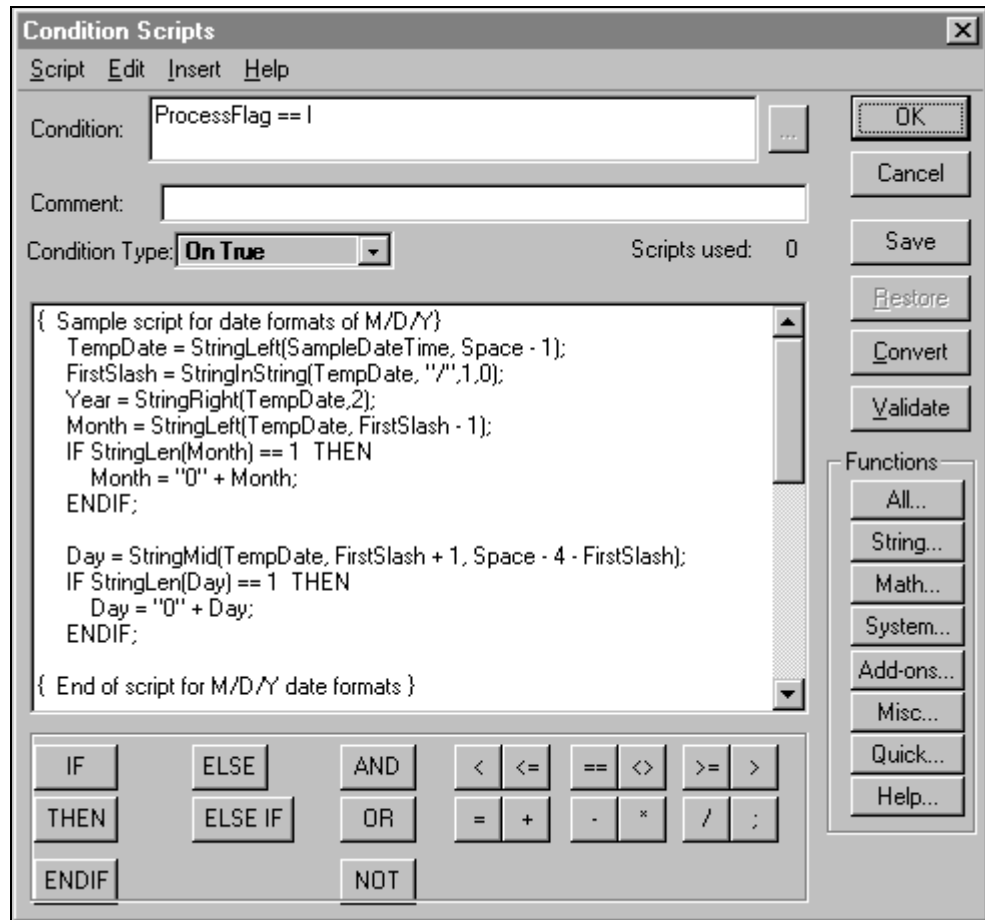
Condition Scripts are linked to a discrete tagname or expression that equates to true or false. You can also use discrete expressions that contain analog tagnames (see example below). There are four types of scripts that you can apply to a condition:

- On True**                      Executes one time when the condition transitions to true.
- On False**                    Executes one time when the condition transitions to false.
- While True**                   Executes continuously while the condition is true.
- While False**                 Executes continuously while the condition is false.

➤ **To access the Condition Script editor:**

1. On the **Special** menu, point to **Scripts**, and then click **Condition Scripts**, or in the Application Explorer under **Scripts**, double-click **Condition**. The **Condition Script** editor appears:

☞ In the Application Explorer under **Scripts**, you can also right-click **Condition**, and then click **Open**.



2. Since Condition Scripts are executed based upon a condition being met, you must specify the condition (a discrete tagname or expression) in the **Condition** box.

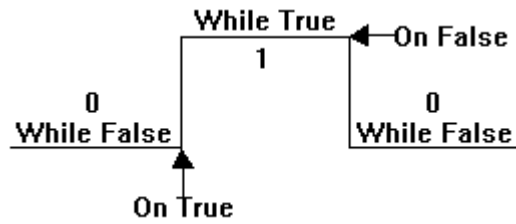
☞ You can also use a discrete expression that equates an analog tagname to true or false. For example, TankLevel >= 75. In this example, when the value of the tagname TankLevel is greater than or equal to 75, the script will execute.

---

**Note** The value for the condition **must transition** to become true or false before the script will execute. For example, if the initial value when WindowViewer starts is true, the value must become false and then true again for an **On True** script to execute.

---

☞ You can apply all four script types to the same condition. Both **While True** and **While False** scripts will begin executing after the specified number of milliseconds have elapsed. To cause immediate execution, create duplicate **On True** and/or **On False** scripts. For example:



3. In the **Comment** box type any miscellaneous comments that you want on file regarding the script.

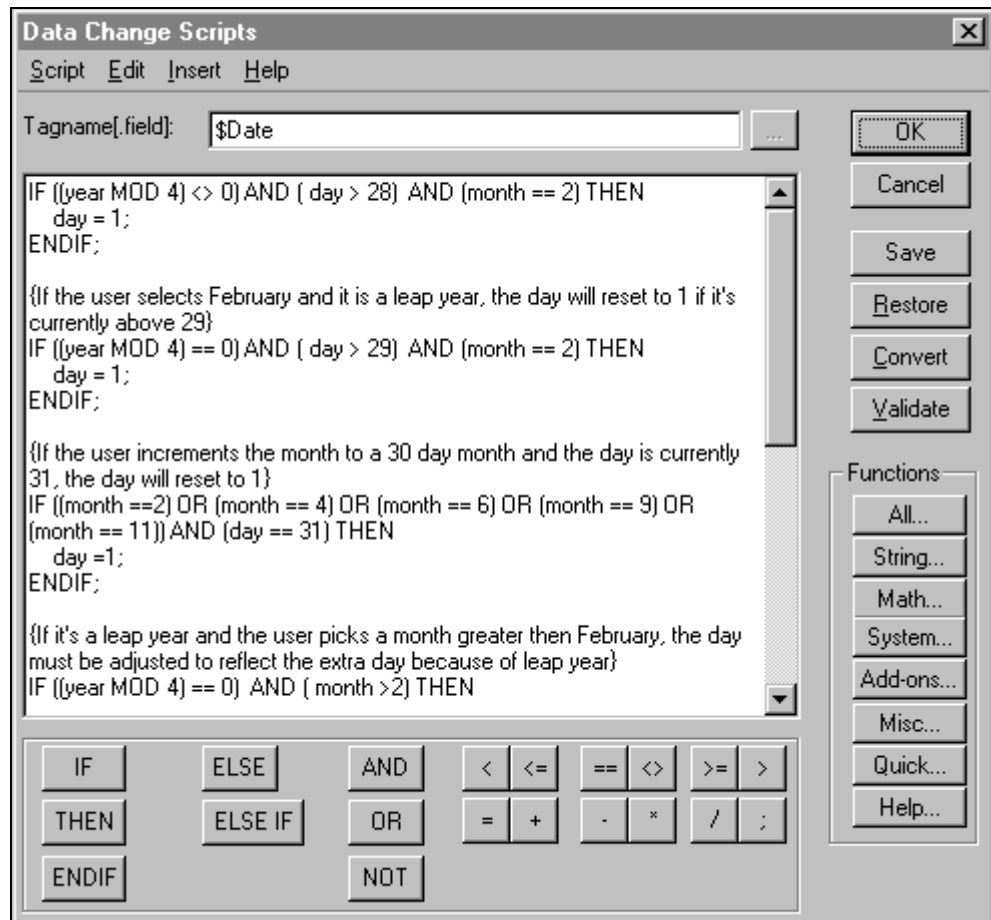


# Data Change Scripts

Data Change Scripts are linked to a tagname and/or **tagname.field** only. They are executed one time when the value of the tagname or **tagname.field** changes by a value greater than the deadband that you defined for the tagname in the Tagname Dictionary.

➤ **To access the Data Change Script editor:**


1. On the **Special** menu, point to **Scripts**, and then click **Data Change Scripts**, or in the Application Explorer under **Scripts**, double-click **Data Change**. The **Data Change Script** editor appears:
  - ☞ In the Application Explorer under **Scripts**, you can also right-click **Data Change**, and then click **Open**.



2. Since Data Change Scripts are executed based upon a change in a data value, you must specify a tagname or **tagname.field** in the **Tagname[.field]** box.
3. On the **Insert** menu, click **Tagname** or double-click the script window. The **Select Tag** dialog box will appear.
  - To select a tagname without a **.field**, double-click the tagname or select it, and then click **OK**. The selected tagname will be inserted into your script at the cursor location.
  - ☞ To quickly access the **Select Tag** dialog box, double-click a blank area in the script editing window. To access a specific tagname's definition in the Tagname Dictionary, type the tagname, and then double-click it.
  - To select a **.field**, first select the tagname that you want to use, then click the **Dot Field** arrow and select the **.field** that you want to associate with the

---

selected tagname. Click **OK**. The selected tagname.**field** will be inserted into your script at the cursor location.

 To quickly insert a tagname **.field**, type the tagname followed by a period (.), and then double-click to the right of the period. The **Choose field name** dialog box will appear. Click the **.field** that you want to use. The dialog box will close and the selected **.field** will automatically be inserted into your script at the cursor location.

 For more information on tagname **.fields**, see your *OpenHMI Reference Guide*.

 For more information on the Tag Browser, see Chapter 3 - Tagname Dictionary.

---

**Important Note** Tagnames that are modified (written to) in a Condition Script or a Data Change Script should **not** be used as the tagname for a Data Change Script or in the expression of a condition script. For example: A Data Change Script that executes on the value of "A" changing, containing the logic "B=B+1." The tagname "B" should **not** be used as the tagname for a Data Change Script or be part of the expression for a Condition Script. Otherwise, it will execute only once - the first time.

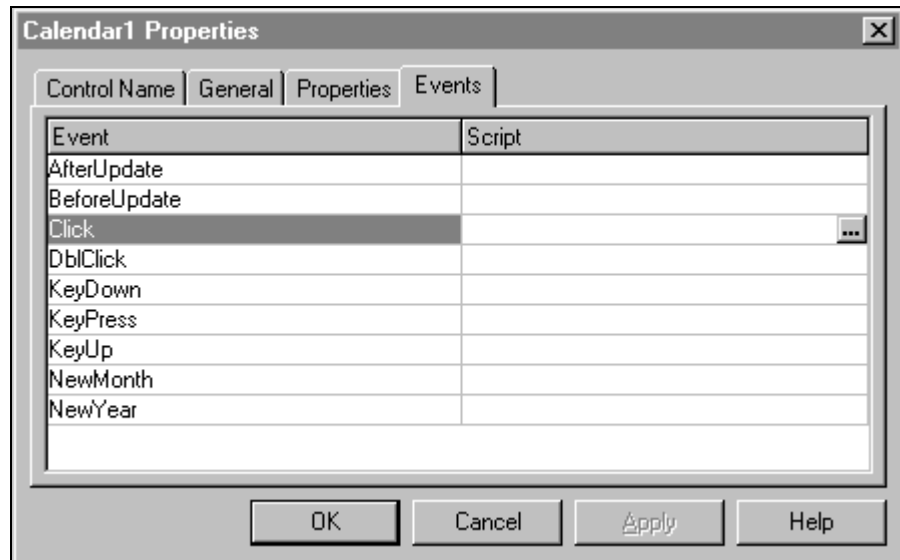
---

## ActiveX Event Scripts

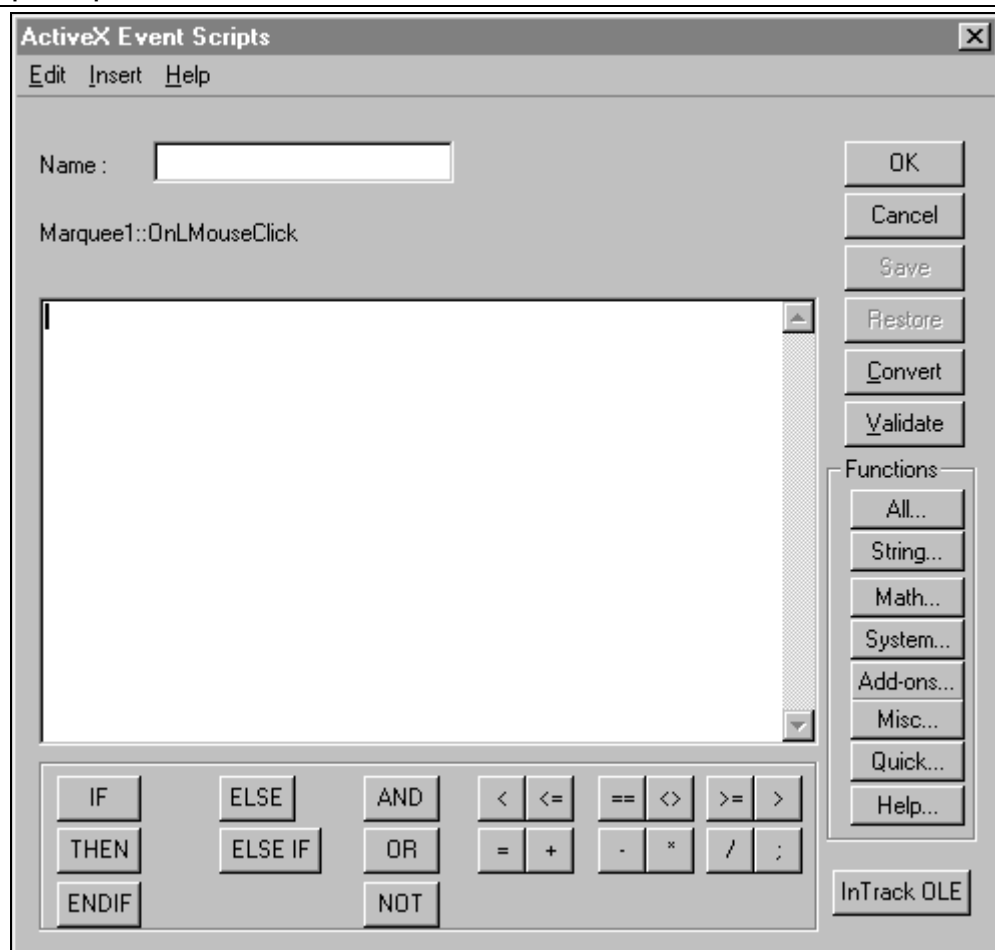
Most ActiveX controls have events associated with them. For example, click, double-click, mouse down and key press are typical events used in many ActiveX controls. OpenHMI ActiveX Event scripts are provided to support event actions. You can associate one ActiveX Event script to each event. You execute ActiveX control events in runtime (WindowViewer).

➤ **To access the ActiveX Event Script editor:**

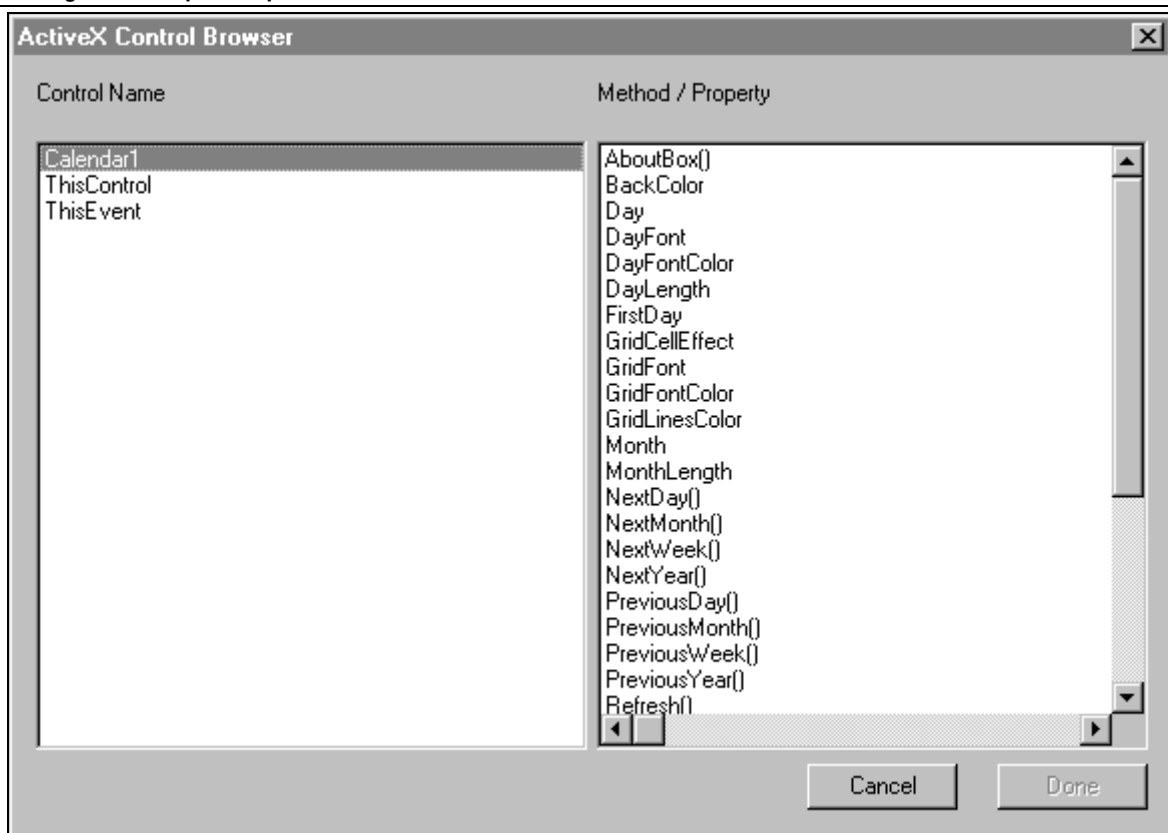
1. Click the **Events** tab in the ActiveX control's **Properties** dialog box to activate the **Events** property sheet. For example:



2. Double-click a blank cell in the **Script** column, or type a name for the ActiveX Event script and click **OK**.
3. If an ActiveX Event script for the name you type does not currently exist, a message box will appear asking you if you want to create it now. Click **OK**. The **ActiveX Event Scripts** editor appears:



4. In the **Name** box, type the name that you want to use to identify the ActiveX Event script.
5. ActiveX control methods are similar to ActiveX control properties. To access the ActiveX control methods, on the **Insert** menu, click **ActiveX**. The **ActiveX Control Browser** appears:



The **ActiveX Control Browser** will display the names of all ActiveX controls being used in your application. When you select a control's name, its respective methods will be displayed. Select the method that you want to insert into your script, and then click **Done**.

## Using Local Variables

You can declare local variables within a script to store temporary results and create complex calculations with intermediate scripting values without impacting or decreasing your licensed tagname count and increase performance.

Local variables or tagnames can be used interchangeably within the same script. However, local variables lose their value and meaning once the script ends where, tagnames are global and retain their values. Unlike tagnames, local variables are declared within the body of the script. The number of local script variables that you can declare within a given script body is limited only by your available memory. Once you have declared a local variable, you can include it in one or more expressions within the same script body. The expression and syntax rules for the placement of local variable names within a script body are the same as those for tagnames, with one exception, local variables do not support **.field** references.

Like tagnames, local variables can be used on both the left and right hand side of statements and expressions that include other local variables and tagnames of different data-types.

### Valid Local Variable Syntax

Each local variable must be declared within the script as a separate **DIM** statement. (One per line cascading is not permitted.) The **DIM** statement syntax and format are as follows:

```
DIM LocalVarName [ AS data-type ] ;
```

Where:

**DIM**

Required keyword.

*LocalVarName*

Variable name that conforms to tagname format and restrictions. Variable names can be up to 32 characters long and must begin with A-Z or a-z. The remaining characters can be: A-Z, a-z, 0-9, !, @, -, ?, #, \$, %, \_ \ and &.

---

**Warning!** If there is a conflict in your script between a declared variable name and a tagname, (both are the same name) the variable name takes precedence over the tagname.

For example, let's assume that you have a tagname defined in your database as "Temp," and you declare "DIM Temp AS Integer;". Within the declaring script, expressions using "Temp" in a statement will refer to the value associated with the local variable "Temp" rather than the tagname "Temp."

---

**AS**

Optional keyword.

☞ If you omit the AS clause from the DIM statement, by default, the variable will be declared as an integer data-type. For example:

```
DIM LocVar1 ;
```

is equivalent to:

```
DIM LocVar1 AS Integer ;
```

---

<i>data-type</i>	Can be any one of the following keywords:
<b>Integer</b>	DIM <i>LocVar1</i> AS Integer;
<b>Real</b>	DIM <i>LocVar2</i> AS Real;
<b>Discrete</b>	DIM <i>LocVar3</i> AS Discrete;
<b>Message</b>	DIM <i>LocVar4</i> AS Message;

The OpenHMI DIM statement cannot be cascaded. For example, the following examples are invalid and cannot be used:

```
DIM LocVar1 AS Integer, LocVar2 AS Real;
```

```
DIM LocVar3, LocVar4, LocVar5, AS Message;
```

To declare the multiple variables in OpenHMI, you must enter a separate DIM statement for each variable. For example, the following examples are valid:

```
DIM LocVar1 AS Integer;
```


```
DIM LocVar2 AS Real;
```

---

### Notes

1. Data-type keywords are case insensitive.
2. The DIM statement line must be terminated with a semi-colon (;).
3. Cascaded DIM statements are not supported.
4. The DIM statement can be located anywhere within the script body. But must precede the first referencing script statement or expression.
5. If the local variable is referenced before the DIM statement, the script editor will read it as a tagname when the script is validated and you will be asked to define it.

---

 For more information on the syntax used for local variables, see your *OpenHMI Reference Guide*.

## Creating FOR-NEXT Loop Scripts

A FOR-NEXT loop is used to perform a function (or set of functions) within a script several times during a single execution of a script. The general format of the FOR-NEXT loop is as follows:

```
FOR AnalogTag = start_expression TO end_expression [STEP
change_expression]
    ...statements...
IF (condition) THEN
    [EXIT FOR;]
ENDIF;
    ...statements...
NEXT;
```

Where:

[ ]	Items enclosed in brackets denote optional parameters.
<b>BOLD</b> CASE	Bold items in <b>UPPERCASE</b> denote script language reserved keywords.
<i>italics</i>	Items in lowercase <i>italics</i> denote variable data.
<i>AnalogTag</i>	An OpenHMI Analog type tagname.
<i>start_expression</i>	A valid OpenHMI expression, to initialize <i>AnalogTag</i> to a value for execution of the loop.
<i>end_expression</i>	A valid OpenHMI expression, if <i>AnalogTag</i> is greater than <i>end_expression</i> , execution of the script jumps to the statement immediately following the <b>NEXT</b> statement. (This holds true if loop is incrementing up, otherwise, if loop is decrementing, loop termination will occur if IntegerTag is less than <i>end_expression</i> .)
<i>change_expression</i>	A valid OpenHMI expression, to define the increment or decrement value of <i>AnalogTag</i> after execution of the <b>NEXT</b> statement.

---

**Note** The *change\_expression* can be either positive or negative. If *change\_expression* is positive, *start\_expression* must be less than or equal to *end\_expression* or the statements in the loop will not execute. If *change\_expression* is negative, *start\_expression* must be greater than or equal to *end\_expression* for the body of the loop to be executed. If **STEP** is not set, then *change\_expression* defaults to 1.

---

<i>...statements...</i>	One or more valid OpenHMI script language statements. These could be nested <b>FOR</b> loops. Nested loops require different <i>change_expressions</i> from outer loops.
<b>FOR</b>	Signals the beginning of the "For" loop.
<b>TO</b>	Signals the beginning of the <i>end_expression</i> .
<b>STEP</b>	Signals the beginning of the <i>change_expression</i> .
<b>EXIT FOR</b>	Immediately terminates the loop with script execution jumping to the statement immediately following the <b>NEXT</b> statement.
<b>NEXT</b>	Signals the end of the loop statement.

When you execute a FOR...LOOP function, OpenHMI:

1. Sets AnalogTag equal to start\_expression.
2. Tests to see if AnalogTag is greater than end\_expression. If so, OpenHMI exits the loop. (If change\_expression is negative, OpenHMI tests to see if AnalogTag is less than end\_expression.)



3. Executes the statements.
4. Increments AnalogTag by 1 - or by change\_expression if it is specified.
5. Repeats steps 2 through 4.

## Nesting FOR-NEXT Loops

FOR-NEXT loops may be nested. Number of levels of nesting possible depend on your system's memory and resource availability.

## Screen Updates and Performance Penalties

During execution of the FOR-NEXT loop, the screen update sub-system within OpenHMI will pause until the loop is complete.

- All animation on the screen will pause during execution of the FOR-NEXT loop. Therefore, you cannot use the FOR-NEXT loop to animate an object to move across the screen, since all movement will occur only after the loop has completed.
- Real-time trends will pause.
- Displayed values will not update on the screen during loop execution. Although there may be new values present within those variables, those new values will only be displayed after the loop has completed.
- The value of any I/O type tagname modified within the body of a FOR-NEXT loop will be transmitted to the I/O server only after loop execution is finished. Therefore, if you modify the value of a I/O type tagname during each interaction of a FOR-NEXT loop, only the final value of that I/O type tagname will be transmitted to the PLC.

---

**Note** FOR-NEXT loops pause all operations in OpenHMI. While executing, no data moves in or out of WindowViewer, no animation links are updated and, no other scripts are executed.

---

## Loop Execution Time Limit

By default, FOR-NEXT loops must complete execution within 5 seconds. This is a safety limit built into the FOR-NEXT sub-system. The time limit will be enforced for all FOR-NEXT loops. For example:

```
FOR X = 1 TO 1000000
  FileWriteMessage("C:\LOG.TXT", "Hello");
NEXT;
```

---

**Note** You can extend this limit by adding the following switch to your OPENHMI.INI file in your application directory:

### **LoopTimeout=20**

Where; 20 is the number of seconds before terminating the loop prematurely.

---

This loop will most likely exceed the time limit of 5 seconds. In the Logger a message will appear that indicates the following:

```
95/03/07 07:34:40.550/VIEW /Exceeded loop time limit of 5
seconds.
95/03/07 07:34:40.550/VIEW /FOR-NEXT Timeout at X = 65464
```

This error message indicates that the FOR-NEXT loop has terminated before meeting the *end condition*, it also provides the value of the loop variable at the time of the loops termination. This information will allow you to track down which FOR-NEXT is having the problem.

**Note** The 5 second time limit is only evaluated each time the NEXT; statement is reached within the FOR-NEXT loop. For example, if you were executing the following script:

```
FOR Index = 1 to 10
    SQLInsert (ConnectionID, "ORG", "list1");
    SQLInsert (ConnectionID, "ORG", "list2");
    SQLInsert (ConnectionID, "ORG", "list3");
    SQLInsert (ConnectionID, "ORG", "list4");
NEXT;
```

If each **SQLInsert()** took 12 seconds to complete, all four inserts would be executed to completion before the loop exited because it had exceeded the 5 second time limit.

## Loop Variable Value After Loop Execution

As in Visual Basic (and most other popular Basic programming languages) the value of the loop variable at the end of loop execution is defined as follows:

The Index continues to increase in value, starting at the *Start\_Condition*, incremented each iteration by the value of *Step\_Expression*, until it reaches the last iteration at which the Index value exceeds that of the *End\_Expression*.

Therefore, if you had a loop as follows:

```
FOR Index = 2 TO 25 STEP 7
    { Some script statements }
NEXT;
```

The value of Index would progress as follows:

Iteration	Value	Calculation
1	9	2 + 7
2	16	2 + 7 + 7
3	23	2 + 7 + 7 + 7
4	30	2 + 7 + 7 + 7 + 7

At that point, when the value reached 30, the loop would stop executing because it exceeded the *End\_Expression*. The ending value of Index would be 30.

## Nested Control Structures

Control structures can be placed inside other control structures (such as an IF...THEN block within a FOR...NEXT loop). A control structure inside another control structure is known as *nested*.

Example:

```
FOR TagX = 1 TO 5
FOR TagY = 1 TO 10
    ..statements...
    IF (condition) THEN
        [EXIT FOR;]
    ENDIF;
    ..statements...
NEXT;
NEXT;
```

Where:

The first NEXT closes the inner FOR loop and the last NEXT closes the outer FOR loop. Likewise, in nested IF statements, the ENDIF statements automatically apply to the nearest prior IF statement.

## Exiting a Control Structure

The EXIT FOR statement allows you to exit directly from a FOR loop. Syntactically, the EXIT FOR statement is simple. EXIT FOR can appear as many times as needed inside a FOR loop:

### Example:

```
FOR TagX = 1 TO 10;  
...statements...  
IF (condition) THEN  
    EXIT FOR;  
ENDIF;  
...statements...  
NEXT;
```

Below are some examples of various FOR-NEXT loop scripts:

### Example 1 "Simple Math 2"

This loop allows a person to configure the number with which to raise to a power, as well as the power to which they would like to raise it by setting up the value input links for the tagnames NumberToRaise and Power:

```
Product = 1;  
NumberToRaise = 4;  
Power = 12;  
FOR Index = 1 TO Power  
    Product = Product * NumberToRaise;  
NEXT;
```

Once the above script has completed processing the value of the **Product** will be 16,777,216.

**Example 2** "Complex FOR-NEXT using indirect tagnames"

This loop utilizes the "EXIT FOR" and "STEP" portions of the FOR-NEXT construct to perform a search on a set of 100 tagnames, searching for the tagname to which NumberEntered is equivalent.

---

**Note** For this example, it is assumed that there are 100 **Memory Integer** tagnames (TAG1 - TAG100) already existing. An operator enters a number into the tagname NumberEntered, and this loop searches TAG1 - TAG100 for a matching value. In addition, there is an indirect analog tagname created: *IndirectTag*

---

```

Found = 0;

FOR Index = 1 TO 100
  IndirectTag.NAME = "TAGNAME" + TEXT( Index, "#" );
  IF (IndirectTag.NAME == ("TAGNAME"+ Text(NumberEntered,"#")))
    THEN
      Found = 1;
      EXIT FOR;
    ENDIF;
NEXT;

IF (Found==1) THEN
  Show "NumberFound"; {window notifying search was successful}
ELSE
  Show "NumberNotFound";
ENDIF;

```

Once the script has completed processing, a window will be displayed either indicating that the number was found, or not.

---

**Note** Notice the use of two addition functions within this script: **Show()** and **TEXT()**.

---

**Example 3**

This loop performs an odd calculation, but illustrates the use of nested FOR-NEXT loops, as well as the use of the "STEP" portion of the FOR-NEXT construct:

```

MyTag = -1;

FOR Index = 1000 TO -1000 STEP -5
  IF (MyTag > Index) THEN
    FOR Index2 = 1 TO 10 STEP 2
      MyTag = MyTag * (Index + 11);
    NEXT;
  ENDIF;
NEXT;

```

Once the script has completed processing the values will be: MyTag = -7776, Index = -1005 and Index2 = 11.

## Script Editing Styles and Syntax

The OpenHMI script editor supports two "styles" of scripts: "Simple" and "Complex." Simple scripts allow assignments, comparisons, simple math functions, and so on. Complex scripts provide the ability to perform logical operations in the form of IF-THEN-ELSE type statements. In addition, OpenHMI also supports the use of built-in complex functions.

An example of a function would be the **StartApp**(*ApplicationName*) function actually started the Windows application identified in the argument, "(*ApplicationName*)". Functions may be used in both Simple and Complex scripts. The following section includes complete descriptions of each style.

## Required Syntax for Expressions and Scripts

The syntax used in scripts and expression dialog boxes is similar to the algebraic syntax of a calculator. Most expressions are assignment statements written in the following form:

```
a = (b - c) / (2 + x) * xyz;
```

This statement would cause the value of the expression to the right of the equal sign (=) to be placed in the variable location named "a." Every expression must end with a semicolon (;). The operands in an expression may be constants or variables. A single tagname must appear to the left of the assignment operator =.

**Memory** or **I/O Message** type tagnames can be concatenated by using the plus (+) operator. For example, tagnames can be concatenated for use in **Indirect** type tagnames. If a Data Change Script such as the one below was created, each time the value of "Number" changed, the indirect tagname "Setpoint" would change accordingly:

```
Number=1;
Setpoint.Name = "Setpoint" + Text (Number, "#" );
```

Where: The result is "Setpoint1."

## Simple Scripts

Simple scripts provide the ability to implement logic such as assignments, math and functions. An example of this type of scripting is:

```
React_temp = 150;
ResultTag = (Sample1 + Sample2)/2;
{this is a comment}
Show "Main Menu";
```

In this example, the script will assign the value of "150" to the tagname "React\_temp." "Sample1" will be added to "Sample2" and the result divided by "2" and the "Main Menu" window will appear on the screen when the script is run.

---

**Note** Notice that each logical statement must end with a semicolon (;) and that several logical statements may be included in one script. Also notice that comments are allowed within the script editor. Comments are identified by a pair of braces {}. The function **Show** was also used with the argument "Main Menu" (WindowName) to cause the specified window to open.

---

In addition to simple assignments, mathematical operators and functions, OpenHMI supports several other "Operations" for use on "Operands," that is, tagnames, number constants, and so on.). **Discrete**, **Integer** and **Real** tagname types are supported for all operations listed below. **Message** tagname types may be used with comparison and assignment operations only. The following is a list of OpenHMI supported operations:

## Operations that Require 1 Operand:

~	Complement
-	Negation
NOT	Logical NOT

## Operations that Require 2 Operands:

*	Multiplication
/	Division
+	Addition and Concatenation
-	Subtraction
=	Assignment
<b>MOD</b>	Modulo
<b>SHL</b>	Left Shift
<b>SHR</b>	Right Shift
<b>&amp;</b>	Bitwise AND
<b>^</b>	Exclusive OR
<b> </b>	Inclusive OR
<b>**</b>	Power
<b>&lt;</b>	Less than
<b>&gt;</b>	Greater than
<b>&lt;=</b>	Less than or Equal to
<b>&gt;=</b>	Greater than or Equal to
<b>==</b>	Equivalency ("is equivalent to")
<b>&lt;&gt;</b>	Not Equal to
<b>AND</b>	Logical AND
<b>OR</b>	Logical OR

## Precedence of Operators

The following list shows the order of precedence for evaluation of operators. The first operator in the list is evaluated first, the second operator is evaluated second, and so on. Operators in the same line in the list have equivalent precedence. Operators are listed from highest precedence to lowest precedence.

( )	Highest Precedence
- , NOT, ~	
**	
*, /, MOD	
+, -	
SHL, SHR	
<, >, <=, >=	
==, <>	
&	
^	
=	
AND	
OR	Lowest Precedence

## Precedence Examples

Since \* has higher precedence than +,

$B + C * D$ ; is equivalent to  $B + (C * D)$ ;

Since \* and / have equivalent precedence,

$B / C * D$ ; is equivalent to  $(B / C) * D$ ;

Some other examples to note:

$B * - D$ ; is equivalent to  $B * (-D)$ ;

B or C and D; is equivalent to B or ( C and D );

## Descriptions of Operators

Arguments for the previously listed operators can be numbers or tagnames. Putting parentheses around the arguments is optional, and the operator names are not case-sensitive.

### Parentheses ( )

Parentheses are used to ensure the correct order of evaluation for the operations. They can also make a complex expression easier to read. Operations in parentheses are evaluated first (preempting the other rules of precedence that would apply in the absence of parentheses). If the precedence is in question or needs to be overridden, use parentheses. In the example below parentheses are used to force B and C to be added together before multiplying by D:

```
( B + C ) * D;
```

### Negation ( - )

Negation is an unary operator that converts a positive integer or real number into a negative number.

### Complement ( ~ )

This operator yields the one's complement of a 32-bit integer. In other words, it converts each zero-bit to a one-bit and each one-bit to a zero-bit. The one's complement operator is an unary operator that accepts an integer operand.

### Power ( \*\* )

This binary operator returns the result of a number (the base) raised to the power of a second number (the power). The base and the power can be any real or integer numbers, subject to the following restrictions:

- A zero base and a negative power are invalid.  
Example: "0 \*\* -2" and "0 \*\* -2.5"
- A negative base and a fractional power are invalid.  
Example: "2 \*\* 2.5" and "-2 \*\* -2.5"

Invalid operands yield a zero result. Moreover, the result of the operation should not be so large or so small that it cannot be represented as a real number. Example:

```
1 ** 1 = 1.0
3 ** 2 = 9.0
10 ** 5 = 100,000.0
```

### Multiplication ( \* ), Division ( / ), Addition ( + ), Subtraction ( - )

These binary operators perform basic mathematical operations. The plus (+) is also used to concatenate **Memory** or **I/O Message** type tagnames. For example, tagnames can be concatenated for use in **Indirect** tagnames. If a Data Change Script such as the one below were created, each time the value of "Number" changed, the indirect tagname "Setpoint" would change accordingly:

```
Number=1;
Setpoint.Name = "Setpoint" + Text (Number, "#" );
Where: The result would be "Setpoint1."
```

### Modulo (MOD)

**MOD** is a binary operator that divides an integer quantity to its left by an integer quantity to its right. The remainder of this division is the result of the MOD operation. Example:

```
97 MOD 8 yields 1
63 MOD 5 yields 3
```

## Shift Left (SHL), Shift Right (SHR)

**SHL** and **SHR** are binary operators that use only integer operands. The binary content of the 32-bit word referenced by the quantity to the left of the operator is shifted (right or left) by the number of bit positions specified in the quantity to the right of the operator. Bits shifted out of the word are lost. Bit positions vacated by the shift are zero-filled. (The shift is an unsigned shift.)

## Bitwise AND ( & )

This is a bitwise binary operator which compares 32-bit integer words with each other, bit for bit. A common use of this operator is to mask a set of bits. The operation in this example would "mask out" (set to zero) the upper 24 bits of the 32-bit word. For example:

```
result = name & 0xff;
```

## Exclusive OR (^) and Inclusive OR (|)

The **ORs** are bitwise logical operators which compare 32-bit integer words to each other, bit for bit. The Exclusive **OR** looks for opposite bit status's in corresponding locations. If the corresponding bits are the same, a zero is the result. If the corresponding bits differ, a one is the result. Example:

```
0 ^ 0 yields 0
0 ^ 1 yields 1
1 ^ 0 yields 1
1 ^ 1 yields 0
```

The Inclusive **OR** examines the corresponding bits for a one condition. If either bit is a one, the result is a one. Only when both corresponding bits are zeros is the result a zero. For example:

```
0 | 0 yields 0
0 | 1 yields 1
1 | 0 yields 1
1 | 1 yields 1
```

## Assignment ( = )

Assignment is a binary operator which accepts integer or real operands. Each statement may contain only one assignment operator. Only one name can be on the left side of the assignment operator. Read the equal sign (=) of the assignment operator as "is assigned to" or "is set to."

---

**Note** Do not confuse the equal sign with the equivalency sign (==) used in IF-THEN-ELSE clauses and relational contacts.

---

## Comparisons ( <, >, <=, >=, ==, <> )

These operators are used in IF-THEN-ELSE statements to execute various instructions based on the state of an expression.

## AND, OR, and NOT

These operators only work on discrete tagnames. Although, if these operators are used on integers or reals, they are *converted* as follows:

**Real to Discrete:** If real is 0.0, discrete is 0, otherwise discrete is 1.

**Integer to Discrete:** If integer is 0, discrete is 0, otherwise discrete is 1.



Thus, if the statement were: "Disc1 = Real1 AND Real2;" and Real1 was 23.7 and Real2 was 0.0, Disc1 would have 0 assigned to it, since Real1 would be converted to 1 and Real2 would be converted to 0.

## Complex Scripts

Complex scripts provide the ability to perform logical operations in the form of IF-THEN-ELSE type scripts and the ability to process loops using FOR-NEXT script structures. The following is an example of an IF-THEN-ELSE script:

```
IF React_temp > 200 THEN
  React_temp_sp = 150;
  PRValve = 1;
  PlaySound("c:\alert.wav",1);
ELSE
  PRValve = 0;
  PlaySound("c:\All_Ok.wav",1);
ENDIF;
```

In this example, the script checks if the reactor temperature is higher than "200." If so, then the "React\_temp\_sp" tagname is assigned the value of "150," the "PRValve" is turned on and the "alert.wav" file is played by calling the **PlaySound()** function. Else, the reactor temperature is lower than "200," the "PRValve" is turned off and the "All\_Ok.wav" file is played.

---

**Note** Notice that each IF statement requires an ENDIF statement. Also be aware that an ELSE statement is not required if unnecessary for the script's functionality. Note the use of the Function **PlaySound(path\_text,number)** in this complex script.

---

## Simple Math

This loop performs a simple iterative mathematical calculation. When executed, Product equal the value of NumberToRaise raised to the power of 10, that is, **Product=NumberToRaise<sup>10</sup>**.

```
Product = 1;
NumberToRaise = 4;
FOR Index = 1 TO 10
  Product = Product * NumberToRaise;
NEXT;
```

Once the script executes, the value of the "Product" will be "1048576."

---

**Note** FOR-NEXT loops pause all operations in OpenHMI. While executing, no data moves in or out of WindowViewer, no animation links are updated and, no other scripts are executed.

---

## IF-THEN-ELSE and Comparisons in Scripts

The IF-THEN-ELSE statement is used to conditionally execute various instructions based on the state of an expression. The following comparison operators are used to set up the conditions in an IF-THEN-ELSE statement:

- < Less than
- > Greater than
- <= Less than or Equal to
- >= Greater than or Equal to
- = Equivalency ("is equivalent to")
- <> Not Equal to

Below are some examples of various complex scripts:

IF-THEN statement with no ELSE clause:

```
IF a <> 0 THEN
```

```
a = a + 100;
```

```
ENDIF;
```

IF-THEN-ELSE statement with one ELSE clause:

```
IF temp > 500 THEN
    Disc = 1;
    Real = 43.7;
ELSE
    Disc = 0;
    Real = 93.4;
ENDIF;
```

IF-THEN-ELSE statement with one ELSE IF clause and no ELSE clause:

```
IF temp > 500 THEN
    Disc = Disc * 10;
ELSE
    IF temp > 250 THEN
        x = y / z;
        a = abc + def;
    ENDIF;
ENDIF;
```

IF-THEN-ELSE statement with one ELSE IF clause and one ELSE clause:

```
IF temp > 500 THEN
    Disc = Disc - 10;
ELSE
    IF temp < 250 THEN
        Disc = Disc + 10;
    ELSE
        Disc = Disc + 50;
        Real = 100;
    ENDIF;
ENDIF;
```

---

**Note** Each IF must have a matching ENDIF and a semicolon must be entered at the end of each statement line.

---

IF-THEN-ELSE statement with multiple ELSE IF clauses and one ELSE clause:

```
IF temp > 100 THEN
    temphihi = 1
    Disc = 50;
ELSE
    IF temp > 80 THEN
        temphi = 1;
    ELSE
        IF temp < 10 THEN
            templo = 1;
        ELSE
            IF temp < 30 THEN
                templolo = 1;
            ELSE
                tempok = 1;
            ENDIF;
        ENDIF;
    ENDIF;
ENDIF;
```

IF-THEN-ELSE statement that tests for Condition 1 *or* Condition 2:

```
IF (pump1 < 50.0) OR (pump2 < 50.0) THEN
    alarm-1 = 1;
ELSE
    alarm-1 = 0;
ENDIF;
```

IF-THEN-ELSE statement that tests for Condition 1 *and* Condition 2:

```
IF (pump1 < 50.0) AND (pump2 < 50.0) THEN
    alarm-2 = 1;
ELSE
    alarm-2 = 0;
ENDIF;
```

IF-THEN-ELSE statement that tests for equivalency:

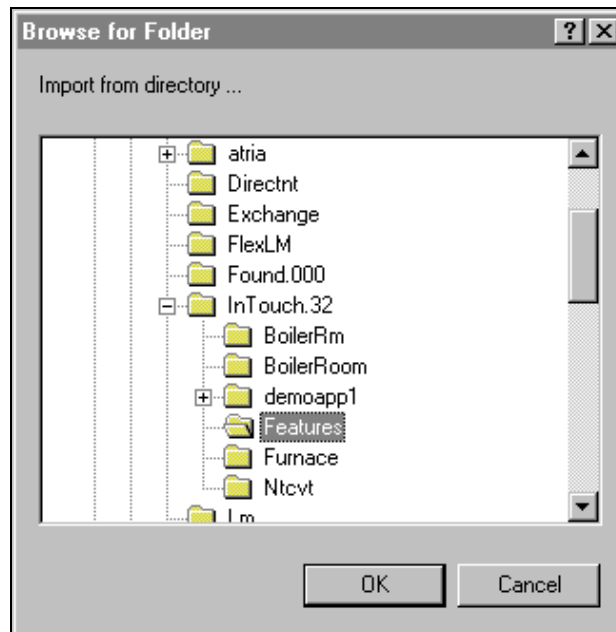
```
IF a > 50 THEN
    IF b == 100 THEN
        c = 0;
    ENDIF;
ENDIF;
```

## Importing QuickScripts

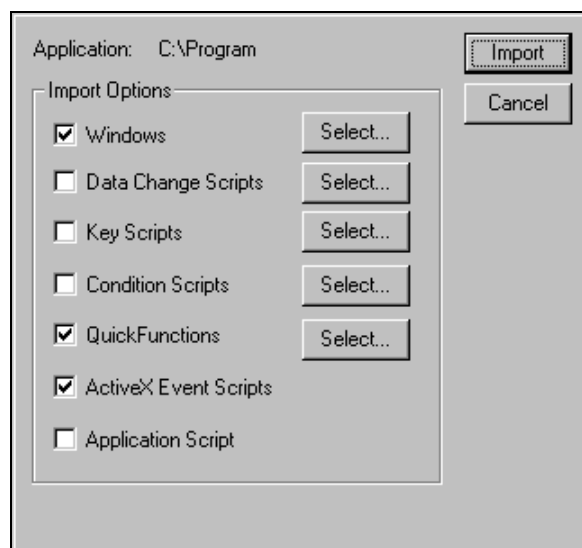
Importing QuickScripts from one OpenHMI application to your current application, can save you a considerable amount of development time. It allows you to reuse your previously created QuickScripts. When you move QuickScripts from one OpenHMI application to another, you must use the **Import** command on the **File** menu.

➤ **To import a QuickScript:**

1. Close all windows in your current application
2. On the **File** menu, click **Import**. The **Browse for Folder** dialog box appears:



3. Locate and select the application directory (folder) containing the QuickScript(s) the you want to import.
4. Click **OK**. The following dialog box appears:



5. Select the QuickScript type(s) that you want to import.
6. Click **Select**. The **Select a ScriptType Script** dialog box appears:



7. Select the QuickScript(s) that you want to import, and then click **OK** to close the dialog box.

---

**Note** When you import ActiveX Event scripts, from one application to another, all ActiveX Events scripts are imported. Additionally, in order for an imported ActiveX Event script to function properly in the new application, the same ActiveX control and the same event for which the script was originally created, must also be used in the new application, and it must be loaded into memory. If the window containing an ActiveX control is closed, its ActiveX Event scripts, or any other OpenHMI QuickScripts containing script functions associated with that ActiveX control, will not execute properly.

---

8. Click **Import**. The system will automatically begin to import the selected QuickScript(s) into your current application.

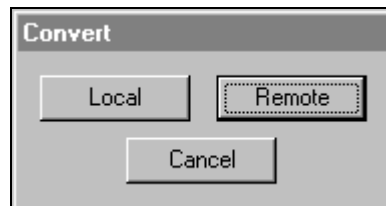
When you import a QuickScript into a new application, all the tagnames in the QuickScript are imported with the QuickScript, but they are not added to your Tagname Dictionary. Instead, they are automatically converted to "placeholder" tagnames. You must convert the placeholder tagnames in order to use them and, if they are not currently defined in the new application's Tagname Dictionary, you will be asked to define each of them.

When the tagnames in an imported QuickScript are converted to placeholder tagnames three index characters are added to the beginning of each tagname. For example, when a discrete tagname is imported, the tagname is prefixed with the three characters **?d:**. When a tagname of 30, 31 or 32 characters in length is imported, the three indexing characters will still be added to the beginning of each tagname. However, the addition of these three characters will not truncate the length of your existing tagname. For example, for placeholder tagnames only, a 32 character tagname is increased to 35 characters. These three additional spaces are allotted only for placeholder tagnames. This increase in tagname length is not supported for standard tagnames.

➤ **To convert placeholder tagnames in an imported script:**

On the **Special** menu, point to **Scripts**, and then click the type of QuickScript you imported, or in the Application Explorer under **Scripts**, double-click the QuickScript type that you imported. The QuickScript editor will appear displaying the first QuickScript on file for the type of script you selected.

1. Click **Convert**. The **Convert** dialog box appears:



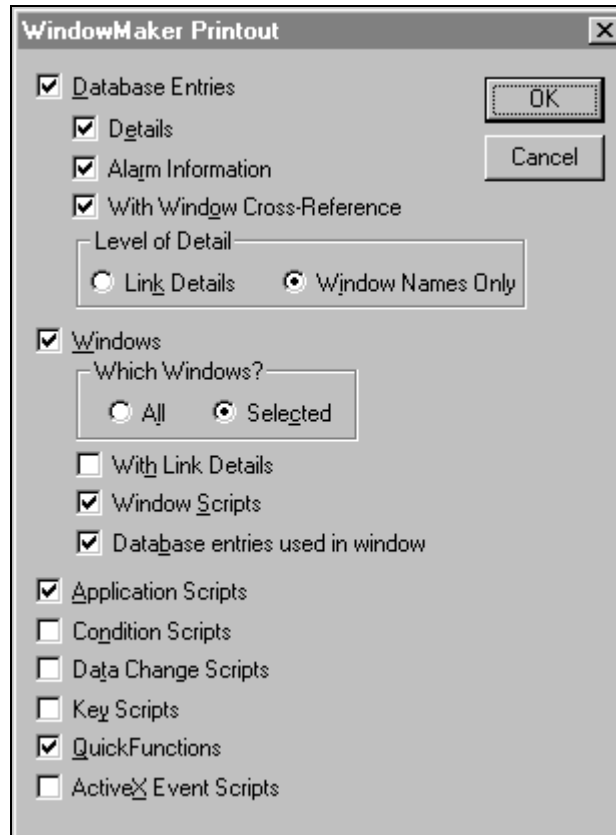
2. Click **Local** to convert the tagnames in the QuickScript to local tagnames.
3. After the tagnames are converted, click **OK** in the QuickScript editor.

# Printing Scripts

You can print all scripts in each OpenHMI QuickScript category.

➤ **To print a script:**

1. On the **File** menu, click **Print**. The **WindowMaker Printout** dialog box appears:



2. To print window scripts, select **Windows**, and then select **Window Scripts**. In the **Which Windows?** group, select **All** to print the scripts for all windows in the application. To print a specific window's script, select **Selected**. The **Windows to Print** dialog box will appear. Select the window(s), whose script you want to print, and then click **OK**.

---

**Note** If you select a window that does not have a script linked to it, the following will be printed on the report: "Window Scripts for *Window Name*: none."

---

3. To print all scripts for a QuickScript type, select the QuickScript type, and then click **OK**.


## Script Functions

OpenHMI provides you with numerous built-in functions that you can be link to objects or pushbuttons or use in scripts to perform a multitude of tasks. For example, acknowledging alarms, hiding windows, changing the tagname being trended by a pen, and so on.

These functions are accessible through the **Insert** menu or by clicking the various buttons in the **Functions** section of the Script Editor. Once you select a function in its respective dialog box, the function and its required arguments are automatically pasted into your script at the cursor location. After the function is pasted into your script, you highlight the argument you want to modify and type in the new value.

## String Functions

String functions are used on string variables. The following briefly describes each string script function.

 For more information on the valid syntax for each function and examples of how you use each function, see your *OpenHMI Reference Guide*.


Function	Description
<b>DText</b>	Changes a message tagname based on the value of a discrete tagname.
<b>StringASCII</b>	Returns the ASCII value of the first character in a specified message tagname.
<b>StringChar</b>	Returns the character corresponding to a specified ASCII code.
<b>StringFromIntg</b>	Converts an integer value into its string representation in another base.
<b>StringFromReal</b>	Converts a real value into its string representation, either as a floating-point number or in exponential notation.
<b>StringFromTime</b>	Converts a time value (in seconds since Jan-01-1970) into a particular string representation.
<b>StringInString</b>	Returns the position in <i>Text</i> where <i>Search For</i> first occurs.
<b>StringLeft</b>	Returns the number of characters specified by Chars starting with the leftmost character of text.
<b>StringLen</b>	Returns the length of text to integer result.
<b>StringLower</b>	Converts all the uppercase characters in text to lower case, and places the resulting sting in MessageResult.
<b>StringMid</b>	Extract from text the specific numbers of characters specified by Chars, starting at the position <i>StartChar</i> . This function is slightly different from its counterparts <b>StringLeft()</b> function and <b>StringRight()</b> function in that it allows the user to specify both the start and end of the string which is to be extracted from the message tag.
<b>StringReplace</b>	Replaces or changes specific parts of a provided string..
<b>StringRight</b>	Returns the number of character specified by Chars starting at the rightmost character of text..
<b>StringSpace</b>	Generates a string of spaces either within a message tagname or an expression..

Function	Description
<b>StringTest</b>	Tests the first character of text to determine whether it is of a certain type..
<b>StringToIntg</b>	Converts the numeric value of a message tagname to an integer value to which mathematical calculations can be applied..
<b>StringToReal</b>	Converts the numeric value of a message tagname to a real (floating point) value to which mathematical calculations can be applied. .
<b>StringTrim</b>	Removes unwanted spaces from text..
<b>StringUpper</b>	Converts all the lowercase character in text to uppercase..
<b>Text</b>	Causes a message type tagname to display the value of an analog (integer or real) tagname based upon the specified <i>Format_Text</i> ..

## Math Functions

Math functions are used on integer or real tagnames. In the following math functions, the **ResultNumericTags** and **InputNumericTags** can be either **Real** or **Integer** and freely intermixed. Keep in mind, however, that returning a non-integer result of a function to an **Integer** tagname will result in the truncation of the result. (The portion to the right of the decimal point will be lost.) The following examples assume that **ResultNumericTag** has been defined as either a **Memory Real** or **I/O Real** type tagname.

The following briefly describes each math script function.

 For complete details on the valid syntax for each function and examples of how you each function, see your *OpenHMI Reference Guide*.

Function	Description
<b>Abs</b>	Returns the absolute value (unsigned equivalent) of a specified number.
<b>ArcCos</b>	Given a number between -1 and 1 (inclusive), returns an angle between 0 and 180 degrees whose <i>cosine</i> is equal to that number.
<b>ArcSin</b>	Given a number between -1 and 1 (inclusive), returns an angle between -90 and 90 degrees whose <i>sine</i> is equal to that number.
<b>ArcTan</b>	Given a number, returns an angle between -90 and 90 degrees whose <i>tangent</i> is equal to that number.
<b>Cos</b>	Returns the <i>cosine</i> of an angle given in degrees.
<b>Exp</b>	Returns the result of <i>e</i> raised to a power.
<b>Int</b>	Returns the next integer less than or equal to a specified number.
<b>Log</b>	Returns the log of a number.
<b>LogN</b>	Returns the values of the <i>logarithm</i> of <i>x</i> to base <i>n</i> .
<b>Pi</b>	Returns the value of <i>Pi</i> .
<b>Round</b>	Rounds a real number to a specified precision.



Function	Description
<b>Sgn</b>	Determines the sign of a value (whether it is positive, zero, or negative).
<b>Sin</b>	Returns the <i>sine</i> of an angle given in degrees.
<b>Sqrt</b>	Returns the <i>square root</i> of a number.
<b>Tan</b>	Returns the <i>tangent</i> of an angle given in degrees.
<b>Trunc</b>	Truncates a real (floating point) number by simply eliminating the portion to the right of the decimal point.

## System Functions

System functions are used to perform actions on your system such as activating another Windows application, copying, deleting or moving files and retrieving information regarding your application. There are two types of system functions; File and Info. The System File functions are used to read and write data from files. They each have two common parameters, *Filename* and *FileOffset*.

The *Filename* refers to the name of the file which will be read from or written to. This filename must include the full path. The *FileOffset* refers to the position in the file where the read or write operation will begin (measured in bytes from the beginning of the file). The first byte of the file is *FileOffset* 0. Upon completion, each function returns the byte position directly following the data that was just read from or written to the file. For example, if the function reads 5 bytes of data starting at byte position 10, the function will return 15.


The *FileOffset* tagname can be used as both a parameter to the functions and as the return tagname. This can facilitate continuous operations.

### Example:

```
FileOffset=FileReadMessage(Filename,FileOffset,Message_Tagname,0);
```

In the previous example, a line of text is read from *Filename*. The starting location is specified by the original value of *FileOffset* (0 for instance, being the beginning of the file). The position where the next read will begin is then returned to *FileOffset* in preparation for the next call to **FileReadMessage()**. Every time the function is called, *FileOffset* gets larger and larger as the **FileReadMessage()** reads through the file.

The following briefly describes each system script function.

 For complete details on the valid syntax for each function and examples of how you use each function, see your *OpenHMI Reference Guide*.


Function	Description
<b>ActivateApp</b>	Activates another currently running Windows application.
<b>FileCopy</b>	Copies a <i>SourceFile</i> to a <i>DestFile</i> , similar to the DOS Copy command or the Copy function in Windows File.
<b>FileDelete</b>	Deletes unnecessary or unwanted files.
<b>FileMove</b>	This is similar to <b>FileCopy()</b> except that it moves the file from one location to another instead of making a copy.
<b>FileReadFields</b>	Reads a Comma Separated Variable (CSV) record from a specified file.

Function	Description
<b>FileReadMessage</b>	Reads a specified number of bytes (or a whole line) from a specified file.
<b>FileWriteFields</b>	Writes a Comma Separated Variable (CSV) record to a specified file.
<b>FileWriteMessage</b>	Writes a specified number of bytes (or a whole line) to a specified file.
<b>InfoAppActive</b>	Tests whether an application is active.
<b>InfoAppTitle</b>	Returns the Application Title or Windows Task list name of a specified program which is currently running.
<b>InfoDisk</b>	Returns information about a specific local (or network) disk drive.
<b>InfoFile</b>	Returns information about a specific file or subdirectory.
<b>InfoOpenHMIAppDir</b>	Returns the current OpenHMI application directory.
<b>InfoResources</b>	Returns various system resource values as follows:  <b>Case 1 and Case 2:</b> GDI and USER are hard-coded to return 50% on Windows NT and Windows 95.  <b>Case 3:</b> On Windows NT and Windows 95 returns "free bytes of paging file."  <b>Case 4:</b> On Windows NT and Windows 95 returns the result of search of all the top level windows. It only counts the windows that are visible and do not have any owners. This is not the actual "number of tasks currently running" in the system. Its closest approximation would be the count of items on <b>Applications</b> tab when you run the <b>Task Manager</b> in Windows NT or the <b>Close Program</b> window which comes up when you press CTRL+ALT+DEL in Windows 95.
<b>StartApp</b>	Automatically starts another Windows application.

## Misc Functions

Miscellaneous functions are used to perform miscellaneous actions such as, hiding a window, printing windows, sending keys to other applications, and so on.

The following briefly describes each miscellaneous script function.

 For complete details on the valid syntax for each function and examples of how you use each function, see your *OpenHMI Reference Guide*.

Function	Description
<b>Ack</b>	Acknowledges any unacknowledged alarm. This function can be applied to a tagname, Alarm Group or Group Variable.
<b>ChangePassword</b>	Displays the <b>Change Password</b> dialog box allowing the logged on operator to change his/her password.
<b>DialogStringEntry</b>	Displays an alphanumeric keyboard on the screen, allowing the operator to change the current string

---

	value of a message tagname in the Tagname Dictionary.
<b>DialogValueEntry</b>	Displays the numeric keypad on the screen, allowing the operator to change the current value of a discrete, integer or real tagname in the Tagname Dictionary.
<b>GetNodeName</b>	Returns the NetDDE node name to a string variable.
<b>GetPropertyD</b>	Retrieves the specified property's discrete value during runtime.

Function	Description
<b>GetPropertyI</b>	Retrieves the specified property's integer value during runtime.
<b>GetPropertyM</b>	Retrieves the specified property's message value during runtime.
<b>Hide</b>	Hides various windows from within a script. A <b>Hide()</b> function must precede the name of each window to be hidden.
<b>HideSelf</b>	Hides the currently active window.
<b>IOSetAccessName</b>	Modifies the <i>application</i> or <i>topic</i> name portions of an Access Name during runtime which allows implementing of hot-backup strategies for OpenHMI.
<b>IOSetItem</b>	Changes the access name and/or item in a I/O type tagname <i>.Reference</i> field.
<b>LogMessage</b>	Writes a user-defined message to the Logger.
<b>PlaySound</b>	Plays a wave form sound specified by a .wav filename or an entry in the <b>[sounds]</b> section of the WIN.INI file through the Windows sound device (if installed).
<b>PrintWindow</b>	Prints the specified window.
<b>RestartWindowViewer</b>	Allows the user control over shutting down and restarting WindowViewer.
<b>SendKeys</b>	Sends keys to an application.
<b>SetDdeAppTopic</b>	This function is replaced by <b>IOSetAccessName</b> beginning with OpenHMI Version 7.0. See <b>IOSetAccessName</b> .
<b>SetDdeItem</b>	This function is replaced by <b>IOSetItem</b> beginning with OpenHMI Version 7.0. See <b>IOSetItem</b> .
<b>SetPropertyD</b>	Specifies the property's discrete value that is to be written during runtime.
<b>SetPropertyI</b>	Specifies the property's integer value that is to be written during runtime.
<b>SetPropertyM</b>	Specifies the property's message value that is to be written during runtime.
<b>Show</b>	Displays a specified window. (Window name must be enclosed in quotation marks.)
<b>ShowAt</b>	Specifies the horizontal and vertical pixel location of a window when it is shown. When the window opens, it will be centered on the horizontal and vertical coordinates.

Function	Description
<b>ShowHome</b>	Displays the "home" window(s). Home windows are those you selected to open automatically when WindowViewer is started. (The home windows are selected in the <b>WindowViewer Properties - Home Windows</b> property sheet.)  <i>☞</i> For more information on home windows, see Chapter 2 - Using WindowMaker.
<b>ShowTopLeftAt</b>	Specifies the horizontal and vertical pixel location of the top left corner of a window when it is shown. When the window opens, its top left corner will be positioned where the horizontal and vertical coordinates meet.
<b>wcAddItem</b>	Adds the supplied string to the list box or combo box.
<b>wcClear</b>	Removes all items from the list box or combo box.
<b>wcDeleteItem</b>	Deletes the item associated with the item index argument in both list or combo boxes.
<b>wcDeleteSelection</b>	Deletes the currently selected item from the list. Applies to list boxes and combo boxes
<b>wcErrorMessage</b>	Given an error number, <b>wcErrorMessage()</b> , returns a string message describing the error. Applies to list boxes, text boxes, combo boxes, radio buttons and check boxes.
<b>wcFindItem</b>	Determines the corresponding index of the first item in the list box or combo box that matches the supplied string.
<b>wcGetItem</b>	Returns the value property of an item string associated with a corresponding index in a list box or combo box.
<b>wcGetItemData</b>	Retrieves the integer value associated with a list item in a list box or combo boxes.
<b>wcInsertItem</b>	Inserts a string into a list box or combo box.
<b>wcLoadlist</b>	Replaces the contents of the list box or combo box with new items.
<b>wcLoadText</b>	Replaces the contents of the text box with a new string.
<b>wcSavelist</b>	Replaces the contents of a filename with the items in a list object.
<b>wcSaveText</b>	Saves the text contained in a text box to a filename.
<b>wcSetItemData</b>	Assigns an integer value to an item in a list box.

## WW DDE Functions

You should not use the WW DDE functions as a replacement for normal OpenHMI DDE communications. Whenever possible, you should create an DDE type tagname to get data from or send data to an external application. The WW DDE functions are intended to support applications that cannot communicate using the DDE Advises supported by OpenHMI. For example, some applications support only DDE Executes or Pokes.

The **WWExecute()**, **WWPoke()** and **WWRequest()** functions use the same Windows functions as MS Visual Basic (DDEML). A single function actually does several things. For example, a **WWPoke()** will perform a DDE Initiate, a DDE Poke and a DDE Terminate all in one function. This makes WW DDE functions less error prone, but also

less efficient in processing many DDE messages. As general guidelines for the use of these functions, you should never:

- Loop these functions (call them over and over).
- Call several of the DDE functions in a row and in the same script.
- Use them to call a lengthy task in another DDE application.

If the DDE command executes a lengthy task in another application it can use up all of the available processor time. However, even if communication difficulties occur, no loss of data will occur. Even if the I/O Server cannot send messages to OpenHMI, it will continue to try.

<b>Function</b>	<b>Description</b>
<b>WWControl</b>	Allows you to Restore, Minimize, Maximize, or Close an application from OpenHMI.
<b>WWExecute</b>	Sends a command (using a DDE Execute ) to a specified <i>Application</i> and <i>Topic</i> .
<b>WWPoke</b>	Pokes a value (using an DDE Poke) to a specified <i>Application</i> , <i>Topic</i> , and <i>Item</i> .
<b>WWRequest</b>	Makes a one-time request for a value (using an DDE Request) from a particular <i>Application</i> , <i>Topic</i> , and <i>Item</i> .

## Script Editor Error Messages

If the script editor detects any errors in the script when it is validated a corresponding message box will be displayed. For example:



☞ In most cases, when an error is encountered, OpenHMI will place the cursor at the position in the script where the error has occurred. However, in some cases, such as a missing ENDIF, the cursor will be at the end of the script. All errors must be corrected before the system will accept the script.

Error Message	Definition
<b>Can only compare alarm groups for equality</b>	Cannot compare alarm groups for <, >, <=, >=.
<b>Cannot add, subtract, multiply or divide with strings</b>	These operations are not supported for strings.
<b>Cannot mix another type with alarm group</b>	Trying to compare an Alarm Group with another type (e.g. integer) or trying to use something other than an Alarm Group somewhere where an Alarm Group is expected.
<b>Cannot mix another type with string</b>	Trying to compare a string with another type (e.g. integer) or trying to use something other than a string somewhere where a string is expected.
<b>Cannot negate alarm groups</b>	Minus sign (-) has been used.
<b>Cannot negate Access Name</b>	A "-" or "~" is not allowed prior to a DDE Access Name.
<b>Cannot negate strings</b>	Minus sign (-) has been used.
<b>Cannot negate TagID</b>	Minus sign (-) has been used.
<b>Cannot negate window</b>	A "-" or "~" is not allowed prior to a window name.
<b>Cannot use Access Name in this manner</b>	A DDE Access Name cannot be used in this context.
<b>Cannot use HistTrendTag in this manner</b>	Trying to compare a string with another type (e.g. integer) or trying to use something other than a string somewhere where a string is expected.
<b>Cannot use TagID in this manner</b>	A TagID type variable cannot be used in this context.
<b>Cannot use window name in this manner</b>	A window name cannot be used in this context.

Error Message	Definition
<b>E Format must be between -38 and +38</b>	The maximum "e" format number is between e-38 and e+38.
<b>E format must have digit following E</b>	Valid "e" format is n.nnen, 1.e is not legal.
<b>Expecting ")" after function arguments for</b>	A matching right parenthesis is required to match the left parenthesis following the function name.
<b>Expecting "(" after function name for</b>	A left parenthesis is required after this function name.
<b>Expecting a number after 0x</b>	Hexadecimal (base 16) numbers can be entered in <b>OpenHMI</b> . To start a hexadecimal number, start with 0x and follow with digits.
<b>Expecting an expression after IF</b>	Missing discrete expression.
<b>Expecting analog argument for function</b>	This argument for this function requires an analog value.
<b>Expecting another argument for</b>	The function requires more arguments than are present.
<b>Expecting another operand</b>	If "a + " is entered, <b>OpenHMI</b> will display this error message.
<b>Expecting assignment</b>	In an action script, a tagname was entered and the next logical operation would be an assignment.
<b>Expecting comma and other argument(s) to function</b>	More arguments are required for this function.
<b>Expecting DLL Name</b>	A DLL name must be used in this context.
<b>Expecting ENDIF</b>	Must have an ENDIF for each IF.
<b>Expecting ENDIF or ELSE</b>	Every IF/THEN must be matched with an ENDIF or ELSE.
<b>Expecting expression after assignment (=)</b>	In an action script, a tagname and assignment were entered and no value was given for the assignment. This can also happen if => is entered instead of >=.
<b>Expecting Function Name</b>	A Function Name must be used in this context.
<b>Expecting name</b>	Expecting a tagname for this argument.
<b>Expecting name in statement</b>	Missing name in statement.
<b>Expecting right parentheses</b>	A matching ")" was not found.
<b>Expecting semicolon</b>	Semicolon is missing at end of line.
<b>Expecting string</b>	The given argument must be a string expression (i.e. the name of a string tagname or a constant string (text surrounded by double quotes ("))).



Error Message	Definition
<b>Expecting THEN</b>	THEN missing after IF statement.
<b>Expecting window name - must be string expression</b>	The given argument must be a string expression (i.e. the name of a string tagname or a constant string (text surrounded by double quotes ("")).
<b>Extra expressions</b>	For example, the expression "a b" is not legal, "a + b" is OK.
<b>Function only legal in action scripts or logic</b>	Some functions are only legal in scripts, not in expressions.
<b>IF expression must be discrete (use == instead of =)</b>	This error is received because of using the assignment (=) instead of comparison (==). For example, "IF a = b THEN ..." should be "IF a == b THEN ...". May also be received if "IF x THEN..." is used and x is not a discrete tagname.
<b>Invalid or missing operand</b>	The operand required for an operator is either invalid or missing.
<b>Invalid placeholder name - must have chars follow ?x:</b>	Describing characters must follow ?x: in placeholder name.
<b>Invalid placeholder name - second char must be d, i, a, r, m, v, g, h, t</b>	Describing 2nd digit character is invalid for placeholder name.
<b>Invalid placeholder name - third char must be ":"</b>	Describing 3rd digit character is invalid for placeholder name.
<b>Logical AND/OR must use discrete</b>	Using AND/OR operators must be done with discrete expressions. Thus "x AND y" is OK, if x and y are discrete tagnames; if they are of any other type, this error message will be received.
<b>Logical NOT must use discrete</b>	Using the NOT operator must be done with discrete expressions. Thus, "NOT x" is OK if x is a discrete tagname, but if x is of any type other than discrete, this error message will appear.
<b>Maximum string 131 characters</b>	The string is longer than the max allowed.
<b>Must assign the return value of function</b>	Certain functions require function evaluation of the return value.
<b>Must have a digit after decimal point</b>	The syntax "1." is not legal.
<b>Must have hist trend variable for this argument</b>	A Hist Trend type variable must be used in this context.
<b>Must have writeable analog variable or name.field for this argument</b>	The argument must be an integer or real variable or integer or real .field of a variable.

Error Message	Definition
<b>Must have writeable discrete variable for this argument</b>	The argument for this function must be a tagname whose type is discrete and for which read-only is NOT checked.
<b>Must have writeable integer variable for this argument</b> and	The argument for this function must be a tagname whose type is integer for which read-only is NOT checked.
<b>Must have writeable message variable for this argument</b>	The argument for this function must be a tagname whose type is message and for which read-only is NOT checked.
<b>Must have writeable real variable for this argument</b>	The argument for this function must be a tagname whose type is real and for which read-only is NOT checked.
<b>Must have writeable variable for this argument</b>	The argument for this function must be a tagname for which read-only is NOT checked.
<b>Must not assign the return value of function</b>	Certain functions do not return a value and a return value cannot be evaluated for them.
<b>Name too long</b>	Tagnames must be $\leq 32$ .
<b>No closing comment</b>	Opening comment delimiters ( { ) must be matched with closing comment delimiters ( } ).
<b>No closing quote</b>	Missing closing quotation mark ( " ).
<b>Not enough room in display buffer</b>	Not enough memory for this operation. Clear up memory and this operation will complete.
<b>Not enough room in expression buffer</b>	Not enough memory for this operation at this time. Clear up memory and this operation could complete.
<b>Number too large</b>	Absolute value of numbers must be $< 2e38$ .
<b>Number too small</b>	Absolute value of numbers must be $> 2e-38$ .
<b>Only 8 digits allowed after 0x</b>	When entering a hexadecimal number, only 8 digits are allowed.
<b>Too many arguments</b>	Supplied more arguments to this function than are necessary.
<b>Trying to assign to a read-only name</b>	It is not legal to assign a value to a tagname for which read-only is checked.
<b>Undefined field name</b>	The .field name is not defined, most likely due to a spelling error.
<b>Unrecognized character</b>	The highlighted character is not a legal character for expressions or action scripts.
<b>Using a reserved field name (e.g., SP) for normal tagname</b>	Can't use a field name for tagname.

## Error Messages for Windows Controls

The Window Controls script functions return a value based on the result of processing the script function. The return value, used for error diagnostics, can be assigned to an Integer tagname. For example:

```
ErrorNumber = wcGetItem("ControlName", Number, Tagname);
```

Where:

ErrorNumber is an **Integer** tagname type that will hold the returned error value. The return value of the function can be passed to the **wcErrorMessage()**. The **wcErrorMessage()** will return a string description of the error. For example:

```
ErrorMsg = wcErrorMessage(ErrorNumber);
```

Where:

ErrorMsg is a **Message** type tagname that holds the text description of the returned error. The following table identifies the numeric value and its description:

Error Message	Description
0	Success
-1	General failure
-2	Insufficient memory available
-3	Property is read-only
-4	Specified item already present
-5	Object name unknown
-6	Property name unknown
-x*	Unknown error

\* -x represents any other number.

## CHAPTER 6

# Alarms/Events

OpenHMI provides a notification system to inform operators of process and system conditions. This system supports the displaying and printing of process alarms and system events. Alarms represent warnings of process conditions, while events represent normal system status messages.

OpenHMI includes a standard alarm system. It is used to display and acknowledge events and alarms generated by the local OpenHMI application. This chapter describes the alarm system, the various types of alarm conditions, and the grouping hierarchies. Specific sections cover how you add, modify and delete Alarm Groups, assign tagnames to Alarm Groups, define the alarm conditions for a tagname, display, log and print alarms, and how you configure both alarm systems.

---

**Note:** OpenHMI runtime workstations do not support distributed alarm features.

---

## Alarms and Events

OpenHMI has two types of notifications to inform operators of process activity: Alarms and Events. Alarms represent warnings of process conditions that could cause problems, and require an operator response. A typical alarm is triggered when a process value exceeds a user-defined limit, such as an analog value exceeding a hi-limit threshold. This triggers an *unacknowledged* alarm state which can be used to notify the operator of a problem. If configured to do so, OpenHMI can also log this alarm to a disk-based file and print it out to a printer. Once the operator acknowledges the alarm, the system returns to an *acknowledged* state.

Events represent normal system status messages, and do not require an operator response. A typical event is triggered when a certain system condition takes place, such as an operator logging into OpenHMI. If configured to do so, OpenHMI can log an event to a disk-based file and print it out to a printer.

You can configure any tagname to do event monitoring while you are defining it in the Tagname Dictionary. When you define a tagname to do event monitoring, an event message is logged to the alarm system each time the tagname's value changes. The event message logs how the value changed, whether the change was initiated by the operator, I/O, scripts or the system.

☞ For more information on configuring a tagname to do event monitoring, see Chapter 3 - Tagname Dictionary.

## Alarm Types

OpenHMI classifies alarms into several general categories based on their characteristics. These categories are known as *Type* and *Class*. The standard alarm system categorizes all alarms into five general *Types*: Discrete, Deviation, Rate-of-Change, Value, and SPC.

You can associate each alarm with an OpenHMI tagname. Depending upon a tagname's type, you can define one or more of the alarm classes or types for it. You define your alarm conditions in the Tagname Dictionary.

☞ For information on defining alarm conditions, see Chapter 3 - Tagname Dictionary.

## Event Types

OpenHMI also classifies events into general categories based on their characteristics. These categories are known as *Event Types*. The table below summarizes the classification:

Event	Condition
ACK	Alarm was acknowledged
ALM	Alarm has occurred
EVT	An alarm event occurred
RTN	Tagname returned from an alarm state to a normal state
SYS	A system event occurred
USER	\$Operator changed
DDE	The tagname value was poked from a DDE client
LGC	A QuickScript modified the tagname value
OPR	The operator modified the tagname value using the Value Input

The first six events listed are configured automatically when event logging is enabled. The remaining three you must define for each tagname in the Tagname Dictionary.

☞ For more information, on events, see "Alarms and Events."

## Alarm Priorities

Each alarm configured in OpenHMI has a priority value associated with it. This value represents the severity of the alarm and can range from 1 to 999 with 1 being the most severe. By creating alarm ranges using these priorities and assigning alarms to each, you can easily filter out critical alarms from non-critical ones. You can also create animation links, acknowledgment scripts, and filtered viewing and printing all based on the priority range.

For example, if a process plant has determined that they need four levels of severity, they could establish ranges as shown below:

<b>Alarm Severity</b>	<b>Priority Range</b>
Critical	0 - 249
Major	250 - 499
Minor	500 - 749
Advisory	750 - 999

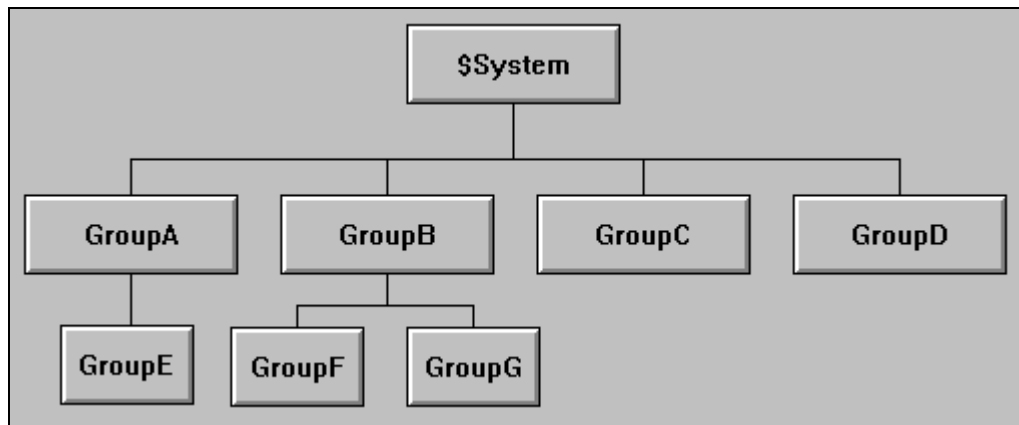
As the plant engineers create OpenHMI tagnames and alarm conditions, each alarm will be assigned to one of these severity levels by choosing a priority number within that range. With these ranges configured, the plant operators may now easily display and print only certain severity levels.

# Alarm Groups

Each OpenHMI alarm is assigned to a logical Alarm Group. These groups are user-definable and can be arranged into a hierarchy up to eight levels deep. The groups provide a way of categorizing alarms based on an organization, plant layout, or any other metric you choose. Alarm Groups are useful for filtering alarm displays, alarm printers, and acknowledgment scripts.

Every tagname is associated with an Alarm Group. If you do not associate an Alarm Group name to a tagname, by default, OpenHMI automatically associates it with the root group, **\$System**. Any Alarm Group may have both tagnames and other Alarm Group names associated with it. Alarm Groups are organized into a hierarchical tree structure with the root group, **\$System**, at the top of the tree. All defined Alarm Groups automatically become descendants of the root group.

This tree may have up to eight levels. Each Alarm Group may have a maximum of 16 subgroups. Each subgroup may have a maximum of 16 subgroups, etc., until the maximum of 8 levels is reached.



This illustration displays only Alarm Groups, not the tagnames within each group. This tree concept is analogous to the MS-DOS directory structure, where a directory may contain other sub-directories (analogous to groups) and file names (analogous to tagnames).

---

**Note** While Alarm Groups do not count as tagnames with OpenHMI licensing, they do count as tagnames in the database. Therefore, the total number of Alarm Groups plus actual tagnames cannot exceed the number of tags allowed by your license.

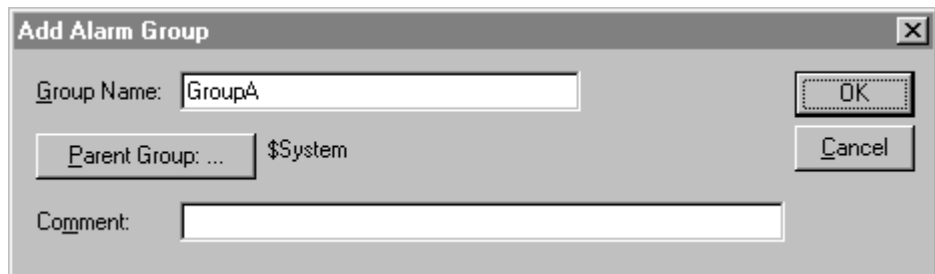
---

➤ **To create an Alarm Group:**

1. On the **Special** menu, click **Alarm Groups**. The **Alarm Group Definition** dialog box appears:
  - ☞ You can also create Alarm Groups and associate tagnames with them while you are defining your tagnames in the Tagname Dictionary.



2. Click **Add**. The **Add Alarm Group** dialog box appears:
  - ☞ The **Modify** and **Delete** buttons are not available until an Alarm Group is defined. The **\$\$System** Alarm Group cannot be modified or deleted.



- ☞ If you right-click a text box in any alarm configuration dialog box, a menu will appear displaying the commands that you can apply to the selected text.
3. In the **Group Name** box, type the name for the new Alarm Group. Since this is the first Alarm Group you have created, it is automatically assigned to the **\$\$System** Parent Group.
    - ☞ After you have created an Alarm Group, it can be used as a Parent Group.
  4. Click **Parent Group** to assign your Alarm Groups, to a different Parent Group. The **Alarm Group Selection** dialog box appears:



5. In the **Select an Alarm Group** list, double-click the name of the Alarm Group that you want to use as the Parent Group (or select it, then click **OK**) for the new Alarm Group. The **Add Alarm Group** dialog box reappears displaying the selected Parent Group. For example:



The 'Add Alarm Group' dialog box has the following fields and controls:

- Group Name:** Text input field containing 'GroupE'.
- Parent Group:** A button labeled 'Parent Group: ...' next to a dropdown menu showing 'GroupA'.
- Comment:** Text input field containing 'GroupE is assigned to parent GroupA'.
- Buttons:** 'OK' and 'Cancel' buttons.

6. In the **Comment** box, type any comment for the new Alarm Group.
7. Click **OK**. The **Alarm Group Definition** dialog box appears displaying your Alarm Group hierarchy:

The 'Alarm Groups' dialog box displays a tree view of alarm groups:

- \$System
  - GroupA
    - GroupE (selected)

Buttons on the right: Close, Add..., Modify..., Delete.

8. Click **Close**.

➤ **To delete an Alarm Group:**

1. On the **Special** menu, click **Alarm Groups**. The **Alarm Group Definition** dialog box appears:

☞ You can also delete Alarm Groups while you are defining your tagnames in the Tagname Dictionary.

This screenshot is identical to the previous one, but the 'Delete' button is highlighted with a dashed border, indicating it is the next step in the process.

2. Select the Alarm Group that you want to delete in the list, and then click **Delete**. A message box will appear asking you to confirm the deletion. Click **Yes** to delete the Alarm Group, or click **No**, to cancel the deletion.

3. Click **Close**.

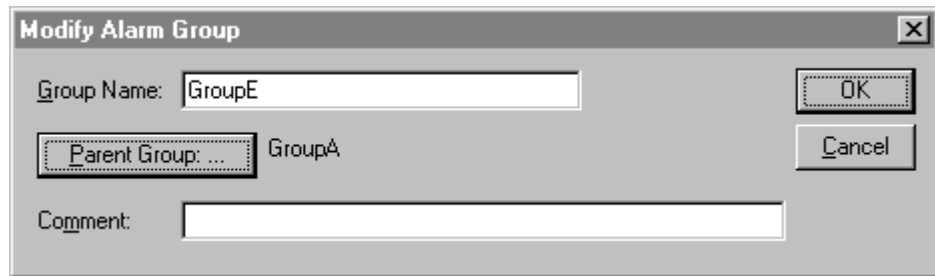
➤ **To modify an Alarm Group:**

1. On the **Special** menu, click **Alarm Groups**. The **Alarm Group Definition** dialog box appears:

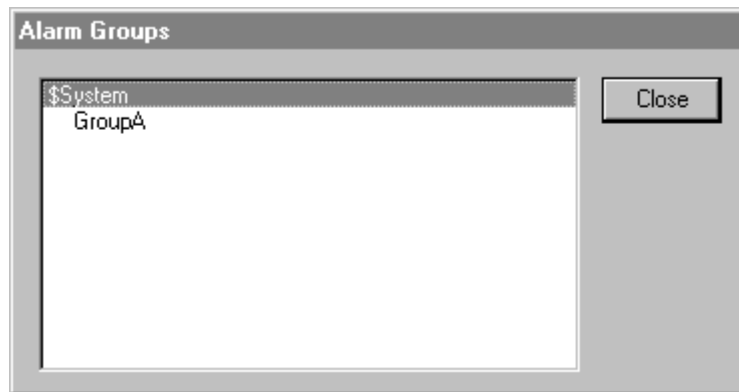
☞ You can also modify Alarm Groups while you are defining your tagnames in the Tagname Dictionary.



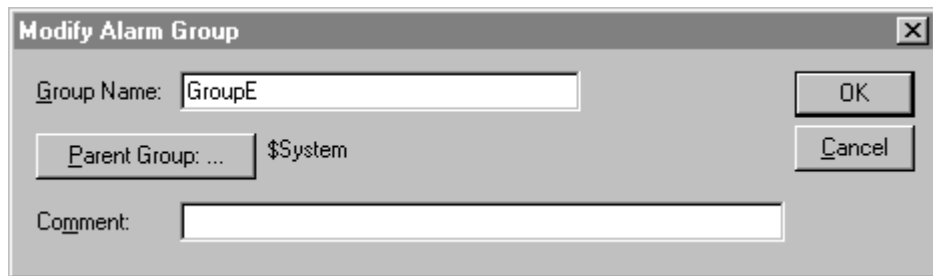
2. Select the Alarm Group that you want to modify in the list, and then click **Modify**. The **Modify Alarm Group** dialog box appears:



3. Make the required changes to the Alarm Group. If you want to change the parent group for the Alarm Group, click **Parent Group**. The **Alarm Groups** dialog box appears:



4. Select the new parent group, and then click **Close**. The **Modify Alarm Group** dialog box reappears displaying the new parent group.



5. Click **OK**.

## Defining Tagname Alarm Conditions

You define alarm conditions in the Tagname Dictionary. Therefore, you can define them at the same time that you are defining the tagname. You can define alarm conditions for discrete type tagnames and analog (integer or real) type tagnames.

*ℳ* For more information on defining alarm conditions, see Chapter 3 - Tagname Dictionary.

## The Standard Alarm Display

The standard alarm system provides you with a unique display object that you use to show locally generated alarms.

The standard alarm display uses two predefined display types: "Alarm Summary" and "Alarm History". The Alarm Summary only displays the current unacknowledged and acknowledged alarms. If an alarm returns to normal (RTN), the entry is removed from the display (if you have configured it to do so). No events are displayed with an Alarm Summary. The Alarm History object displays all of the alarm and events that have occurred. The Alarm History display shows the occurrence of the alarm, the time of acknowledgment (if any) and the time the alarm condition returned to normal.

☞ For more information, see the "Configuring the Standard Alarm System."

In both the Alarm Summary and the Alarm History display objects, each entry is shown as a separate line. The number of entries displayed is determined by the size you have drawn the object and the size of the font that you are using. The standard alarm display lists all active alarms or subsets of active alarms as determined by the current value of the Alarm Group and priority expression associated with the particular alarm display.

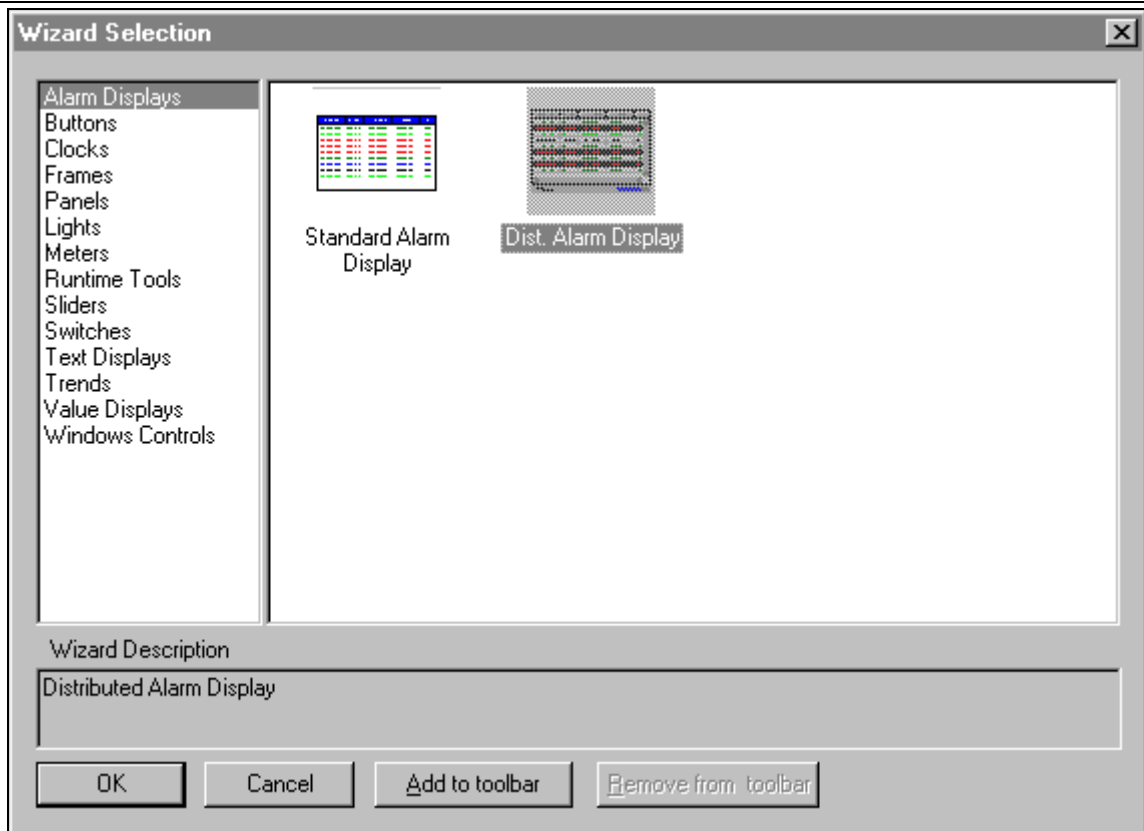
OpenHMI allows you to configure how many alarms are stored for the Alarm History object, the appearance of the alarm displays including, the information that is displayed, logged and printed. You can also select the colors used for the title bar, title text, the background of the alarm display, and the colors used to display the various alarm conditions in the window. In addition, the Alarm Group and alarm priority levels displayed can be dynamically controlled at runtime.

## Creating a Standard Alarm Display

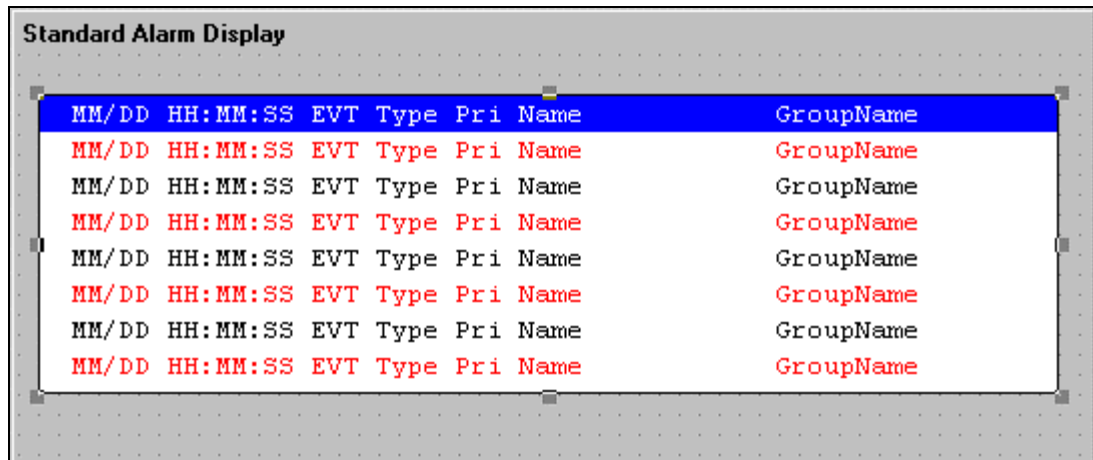


➤ **To create a standard alarm display:**

1. Click the wizard tool in the **Wizard/ActiveX Toolbar**. The **Wizard Selection** dialog box appears:



2. Select the **Alarm Displays** category in the list of wizards to display both alarm wizards.
3. Double-click the **Standard Alarm Display** wizard or select it, and then click **OK**. The dialog box closes and your window reappears with the cursor in the "paste" mode. Click in the window to paste the alarm display:



☞ An alarm display object is like any other object drawn in WindowMaker. It can be moved by grabbing it with the mouse or it can be resized by grabbing one of the object "handles." The text that appears when the alarm display is initially drawn is sample text to show you how the actual display will look at runtime. You can place multiple alarm displays in a window.

4. You are now ready to configure the alarm display object.

## Configuring a Standard Alarm Display

The first time you paste an alarm object, the system default configuration settings are used. Once you have configured an alarm object, the next one you create will, by default, be configured with the same settings.

➤ **To configure a standard alarm display object:**

1. Double-click the alarm display or, with the alarm display selected, on the **Special** menu, click **Animation Links**. The **Alarm Configuration** dialog box appears:

☞ To quickly access the dialog box, right-click the alarm display object, and then click **Properties**.

☞ If you right-click a text box in any alarm configuration dialog box, a menu will appear displaying the commands that you can apply to the selected text.

2. Select the **Window Type** for the alarm display that you want to create:
  - Alarm Summary** - Displays a summary of all of the currently active alarms
  - Alarm History** - Displays a history of alarm events.
3. Select **Titles** if you want to display a title bar with labels for each column on the alarm display. (Selecting this option activates the **Title Bar Color** and **Title Text Color** selection boxes.)
4. Click the **Title Bar Color** box to open the OpenHMI color palette. Click the color in the palette that you want to use for the object's title bar.
  - ☞ Repeat this process for all displayed color selections.
5. Select the **Display Alarms** type that you want to use:
  - Local** - To display locally generated alarms and events.
  - Server** - To display the alarms/events collected by the server node.

☞ You define the server node in the **WindowViewer Properties - General** property sheet.

To quickly access the **WindowViewer Properties**, in the tree view pane, under **Configure**, select **WindowViewer**.

☞ For more information on remote alarming, see "Using the Standard Alarm System for Remote Alarming."

- If you want to specify a specific group of alarms that you want to be logged, in the **Alarm Group** box, type an Alarm Group name or the name of a Group Variable. If you want to log all Alarm Groups, type **\$\$System**.

---

**Note** A Group Variable is a tagname that is defined as a **Group Var** type with the name of an Alarm Group assigned to it.

---

☞ If you want to control the choice of alarms logged at runtime, create a Group Variable type tagname, for example, **ALARMGRP**, then configure a key or action button script to assign a specific Group Name to the Group Variable. For example, the following would be entered in the QuickScript:

```
ALARMGRP.Name="AlarmGroupName";
```

- In the **From Priority** and **To Priority** boxes, type the highest and lowest alarm priority level for the range of priorities that you want shown in the alarm display.

---

**Note** The lower the number, the higher the priority of the alarm. For example, 1 is the highest priority. If you use the default values, all priorities will be used.

---

☞ You can type an analog tagname or an expression in either of the priority boxes if you want a tagname's value to determine the priority level to be logged. You can control the alarm priority level being logged by assigning a value to this tagname through an analog input link or QuickScript.

- In both the **Previous Page** and **Next Page** boxes, type the discrete tagname that you want to use to page up and down through the list of alarm messages when there are more alarms than the window can display.
- Click **Select Display Font** to open the **Font** dialog box to choose the font, its style and size that you want used for the messages displayed in the alarm display object.

☞ The standard alarm display requires the use of fixed-pitch fonts (as opposed to variable-pitch or proportionally spaced fonts). This allows the entries to maintain their column format appearance in the display. Therefore, only the fixed-pitch fonts will appear as possible selections for the alarm window.

- Click **Format Alarm Message** to configure the various items you want shown for each alarm message in the alarm display.

☞ For more information on formatting alarm messages, see "Standard Alarm/Event Message Format."

☞ After you format your alarm messages, check your display object to make sure that you have drawn it large enough to show all of your choices. If it is not big enough, the text at the far right side will be truncated.

- Click **OK** to save your settings and close the **Alarm Configuration** dialog box.

## Previous Page and Next Page Buttons

### ➤ To create alarm object page up button:

- Create an object such as a 3-D button.
- Double-click the object and select **Discrete Value Pushbutton** in the animation link selection dialog box. The **Pushbutton -> Discrete Value** dialog box appears:

Pushbutton -> Discrete Value

Tagname:

Key equivalent

Ctrl  Shift  None

Action

Direct  Reverse  Toggle  Reset  Set

OK  
Cancel  
Clear

3. In the **Expression** box type the discrete tagname that you typed in the **Previous Page** box when you configured your alarm object.
4. In the **Action** group, select **Reset**.
5. Click **OK**.
  - ☞ Whenever the value of the discrete tagname transitions from On (1, True) to Off (0, False), the alarm display object will display the previous page. Once the previous page is displayed, the discrete tagname will automatically reset to On (1, True), unless the top of the list has been reached. In this case, the value of the variable remains Off and the previous page action is disabled.

You can also add a visibility link on the button to hide it when the value of the discrete tagname is 1 (Off).
6. To create a "Next Page" button, repeat this procedure using the discrete tagname that you enter in the **Next Page** box when configuring the alarm object.

## Standard Alarm/Event Message Format

The information shown in an alarm display object, logged to disk or printed is configurable. This configuration process involves selecting what information you want displayed and, in some cases, how many characters you want to display.

- ☞ The order of appearance of the items in the message is fixed and cannot be altered.
- **To format alarm messages:**
1. Double-click the alarm display or right-click it, and then click **Properties**. The **Alarm Configuration** dialog box will appear.
  2. Click **Format Alarm Message**. The **Format Alarm Message** dialog box appears:



**Format Alarm Message**

Date     MM/DD     MMM DD     MM/DD/YY     MMM DD YYYY  
 DD/MM     DD MMM     DD/MM/YY     DD MMM YYYY

Time     24 Hour     AM/PM     HH     MM     SS     MSec

Event    (ACK,RTN,ALM,EVT)

Alarm Type    (HIHI,SDEV,OPR,...)

Operator    Length:

Priority

Comment    Length:

Tagname    Length:

Group Name    Length:

Value    Length:

Limit    Length:

Alarm State    (UNACK\_ALM,ACK\_ALM,etc.)

---

MM/DD HH:MM:SS EVT Type Pri Name    GroupName    Value/Limit

☞ If you right-click a text box in any alarm configuration dialog box, a menu will appear displaying the commands that you can apply to the selected text.

**Note** The preview area (at the bottom of the dialog) displays an example of the alarm message as currently configured. This example will show the message using the font selected, but not color.

3. Select **Date** if you want to display the date in the alarm message, and then select the format for the date as follows:

Selection	Display	Selection	Display
<b>MM/DD</b>	02/28.	<b>MM/DD/YY</b>	02/28/97
<b>DD/MM</b>	28/02.	<b>DD/MM/YY</b>	28/02/97
<b>MMM DD</b>	Feb 28.	<b>MMM/DD/YYYY</b>	Feb 28 1997
<b>DD MMM</b>	28 Feb.	<b>DD/MMM/YYYY</b>	28 Feb 1997

4. Select **Time** if you want to display the time in the alarm message, and then select the format for the time as follows:

<b>24 Hour</b>	Selects the 24 hour military time format. For example, three o'clock in the afternoon is displayed as 15:00.
<b>AM/PM</b>	Selects the AM/PM format. For example, three o'clock in the afternoon is displayed as 3:00 PM.
<b>HH</b>	Displays the hour the alarm/event occurred.
<b>MM</b>	Displays the minute the alarm/event occurred.
<b>SS</b>	Displays the second the alarm/event occurred.
<b>MSec</b>	Displays the millisecond the alarm/event occurred.

5. Select **Event** if you want to display the event type. The event types are:

<b>ACK</b>	Displayed when the alarm has been acknowledged.
<b>RTN</b>	Displayed when the alarm condition returns to normal.
<b>ALM</b>	Displayed when the tagname is in alarm state.
<b>EVT</b>	Displayed when the tagname's value is changed more than the deadband by either the operator, I/O, a QuickScript or the system.

6. Select **Alarm Type** if you want to display the alarm type. The alarm types are:

<b>HHI</b> , etc.	Displayed for Alarm Value conditions.
<b>SDEV</b>	Displayed for Minor Deviation Alarm conditions.
<b>LDEV</b>	Displayed for Major Deviation Alarm conditions.
<b>OPR</b>	Displayed if an operator change caused the alarm condition.
7. Select **Operator** if you want to display the logged-on operator's ID associated with the alarm condition. Enter a value in the Length field to control the number of characters displayed (16 characters maximum).
8. Select **Priority** if you want to display the alarm priority.
9. Select **Comment** if you want to display the tagname's comments. These comments were typed in the **Comments** box when the tagname was defined in the database. In the **Length** box, type the number of characters that you want displayed (50 characters maximum).
10. Select **TagName** if you want to display the tagname. In the **Length** box, type the number of characters that you want displayed (32 characters maximum).
11. Select **Group Name** if you want to display the Alarm Group name. In the **Length** box, type the number of characters that you want displayed (32 characters maximum).
12. Select **Value** if you want to display the value of the tagname when the alarm occurred. In the **Length** box, type the number of characters that you want displayed.
  - ☞ The value should be large enough to provide the desired level of precision (15 characters maximum).
13. Select **Limit** if you want to display the alarm limit value of the tagname. In the **Length** box, type the number of characters that you want displayed.
  - ☞ The size of this field should be large enough to provide the desired level of precision (32 characters maximum).
14. Select **Alarm State** if you want to display the state (unacknowledged, acknowledged, etc.) of the alarm.

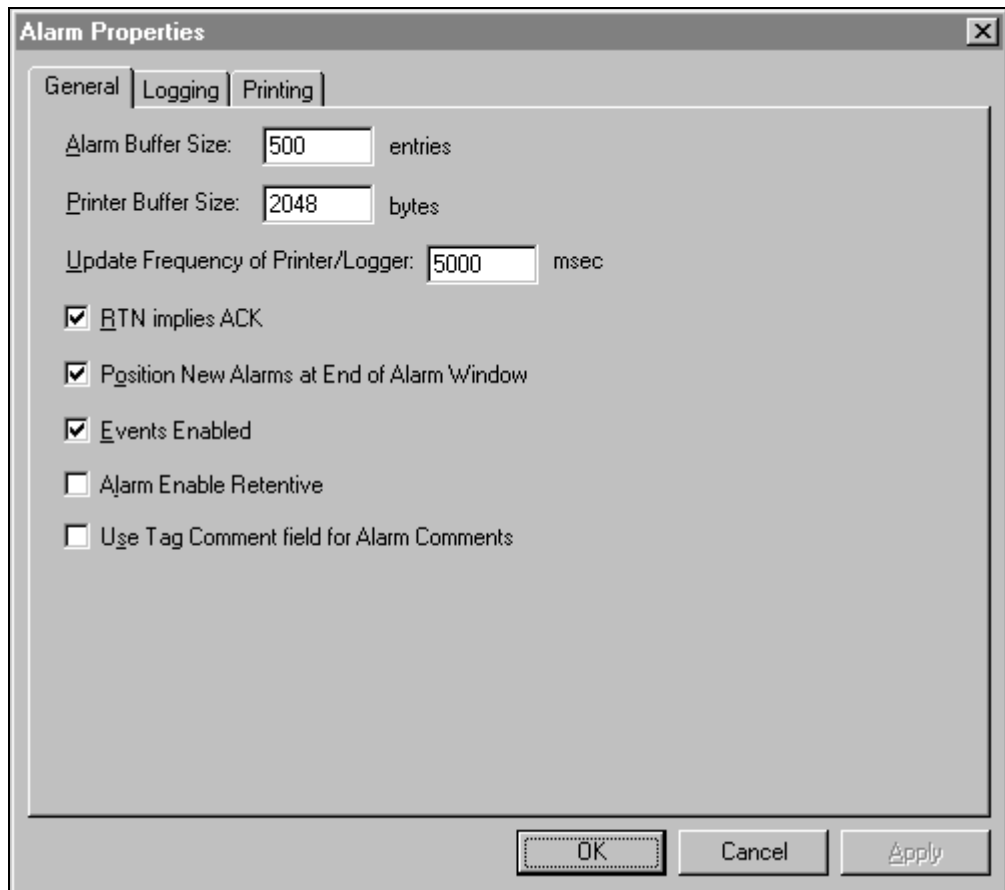
## Configuring the Standard Alarm System

You can configure various parameters for the alarm system, such as the printer and logger buffer size, whether to enable events, the position of new alarms in the alarm display and so on.

**Note** The configuration dialog box behaves like any standard Windows property sheet in that no settings are recorded until you click **OK**. The options are verified for proper entries, however, when you change from one property sheet (tab) to another, if an entry verification fails, the property sheet containing the failed entry is brought back into focus, and a message box appears indicating the error. If you click **Cancel**, all input is ignored and the dialog box closes.

### Alarm/Event General Properties

- **To configure alarms/events general properties:**
  1. On the **Special** menu, point to **Configure**, and then click **Alarms**, or in the Application Explorer under **Configure**, double-click **Alarms**. The **Alarm Properties** appears with the **General** properties sheet active:



☞ If you right-click a text box in any alarm configuration dialog box, a menu will appear displaying the commands that you can apply to the selected text.

2. In the **Alarm Buffer Size** box, the number of "in-memory" alarm events you want WindowViewer to maintain. (The maximum number of alarms that the node can store for summary or history queries.)

Only "in-memory" alarm events can be displayed in alarm display objects. If alarms are not being used, this value may be set to 1 to conserve memory.

**Note** If you set this value too high, it can slow down the performance of your system. If you are using the standard alarm system, a value of 500 is recommended.

In the **Printer Buffer Size** box, type the number of bytes of the buffer that will be used by WindowViewer for parallel printers.

If you are using a serial printer for alarm printing, this entry will have no effect. Only increase the default number (2048) when experiencing problems with printer overflow.

4. In the **Update Frequency of Printer/Logger** box, type the number of milliseconds that WindowViewer wait before trying again to print alarm messages when a printer is offline.
5. Select **RTN implies Ack** if you want alarmed tagnames that return to the "normal" state (RTN) to automatically be acknowledged (ACK). Do not select this option if you want the operator to acknowledge an alarm after it returns to normal.
6. Select **Position New Alarms as End of Alarm Window** if you want new alarms to be displayed at the end of the alarm display object. Alarm windows provide the ability to page back and forth through the alarm queue. Therefore, enabling this option will cause the alarm display object to automatically scroll forward to show the new alarm. If this option is not enabled, the new alarm will be added to the bottom of the list, but the alarm display object will only scroll forward by one line.
7. Select **Events Enabled** if you want to turn on event logging of all data changes that are initiated by the operator, I/O, QuickScripts or, the system. (Only tagnames with **Log Events** selected will be effected.)  
 For more information on events, see "Alarms and Events."
8. Select **AlarmEnable Retentive** if you want the state of the **.AlarmEnable** variable to be retained when WindowViewer is closed.
9. Click **OK** to save your settings and close the dialog box.

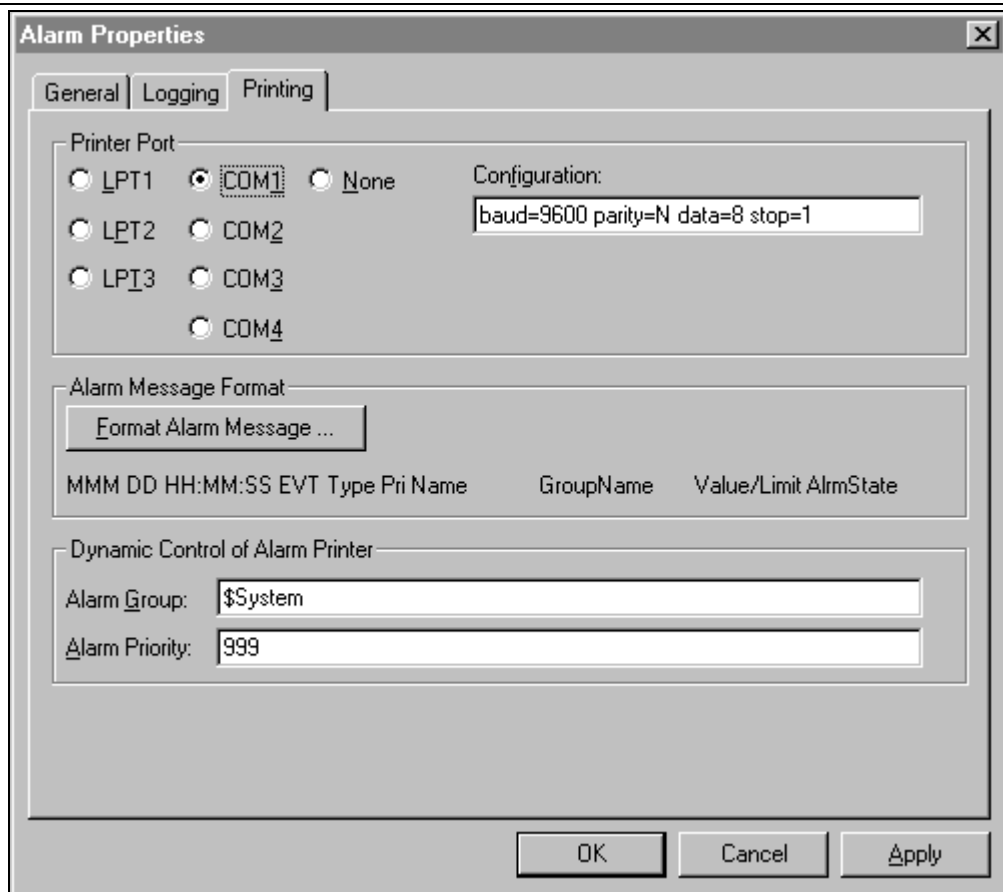
## Alarm Printing Properties

In addition to displaying, OpenHMI also allows you to print alarms. You configure various parameters for alarm printing.

When you are printing alarms, OpenHMI takes complete control of the port. Therefore, a dedicated printer is required. You cannot perform any other printing functions until you stop alarm printing. Because your alarms printouts are limited in configuration (fonts, point size, an so on), a dot-matrix type printer should be sufficient for alarms printing.

### ➤ To configure alarms/events printing:

1. On the **Special** menu, point at **Configure**, and then click **Alarms**, or in the Application Explorer under **Configure**, double-click **Alarms**. The **Alarm Properties** dialog box will appear.
2. Click the **Printing** tab to activate the **Printing** property sheet:



☞ If you right-click a text box in any alarm configuration dialog box, a menu will appear displaying the commands that you can apply to the selected text.

3. Select the **Print to** port being used by the printer.
4. If your printer is using a serial port (COM1-COM4), in the **Configuration** box, type the values for the baud rate, parity, data bits and stop bits. For example:

**baud= 9600 parity=N data=8 stop=1**

The valid values for each entry in the string:

baud 110, 150, 300, 600, 1200, 4800, 9600, 19200

parity O (odd), E (even), N (none)

data 7 or 8

stop 1 or 2

5. If you want to configure the contents of the alarm message that is written to the log file, click **Format Alarm Message**. The **Format Alarm Message** dialog box will appear.

---

**Note** Displayed, printed and logged alarm messages are all formatted in the same manner.

---

☞ For more information on configuring the alarm message format, see "Standard Alarm/Event Message Format."

6. If you want to specify a specific group of alarms that you want to be logged, in the **Alarm Group** box, type an Alarm Group name or the name of a Group Variable. If you want to log all Alarm Groups, type **\$System**.

---

**Note** A Group Variable is a tagname that is defined as a **Group Var** type with the name of an Alarm Group assigned to it.

---

- ☞ If you want to control the choice of alarms logged at runtime, create a Group Variable type tagname, for example, **ALARMGRP**, then configure a key or action button script to assign a specific Group Name to the Group Variable. For example, the following would be entered in the QuickScript:

```
ALARMGRP.Name="AlarmGroupName";
```

7. In the **Alarm Priority** box, type the lowest priority level that you want to be logged. Valid entries are 1 to 999 with 1 being the highest priority. To log all priorities at all times, enter 999 (lowest priority). This will cause all alarms with a priority of less than or equal to 999 to be written to the log file.
  - ☞ You can type an analog tagname or an expression if you want a tagname's value to determine the priority level to be logged. You can control the alarm priority level being logged by assigning a value to this tagname through an analog input link or QuickScript.
8. Click **OK** to save your settings and close the dialog box.

## Alarm .Fields

OpenHMI provides various alarm ".fields" that allow you to dynamically control and/or monitor various alarm conditions. Many of these .fields are accessible using I/O, expressions and/or scripts. I/O access provides the ability to monitor and/or control a specific tagname's alarm information using other Windows applications, for example, Excel, or a remote View application (described later in this chapter).

For example, if you create an analog alarmed tagname called **Analog\_Tagname**, it will have "attributes" associated with it such as its name, its **HiHi** setpoint, and so on. Some of these "attributes" are accessible through logic scripts, expressions and user inputs and are known as **.fields** (dot fields).

The syntax required to access these data fields associated with a tagname is **Tagname.field**. For example, if you want to allow runtime changes to the **HiHi** alarm limit on **Analog\_Tagname**, you could create an **Analog - User Input** touch link could be applied to a button and **Analog\_Tagname.HiHiLimit** would be entered as the expression in the link's dialog box. During runtime, the operator would simply click on the button and type in a new value for the **HiHi** alarm limit being used for **Analog\_Tagname**.

The following briefly describes each examples of how to use the alarm **.fields**, see your *OpenHMI Reference Guide*.

<b>.Field</b>	<b>Description</b>
<b>.Ack</b>	Monitors/controls the alarm acknowledgment status.
<b>.Alarm</b>	Signals that an alarm condition exists.
<b>.AlarmDevDeadband</b>	Monitors/controls the deviation percentage deadband for both minor and major deviation alarms.
<b>.AlarmEnable</b>	Disables/enables events and alarms.
<b>.AlarmValDeadband</b>	Monitors/controls the value of an alarm's deadband.
<b>.DevTarget</b>	Monitors/controls the target for minor and major deviation alarms.
<b>.HiLimit, .HiHiLimit, .LoLimit, .LoLoLimit</b>	Read/write analog tagname .fields that monitors/controls the limits for value alarm checks. These .fields are only valid for integer and real tags.
<b>.HiStatus, .HiHiStatus, .LoStatus, .LoLoStatus</b>	Read only discrete tagname .fields that determines whether an alarm of a specified type exists.
<b>.MajorDevPct</b>	Read/write integer tagname .field that monitors or controls the major percentage of deviation for alarm checking.
<b>.MajorDevStatus</b>	Read only discrete tagname .field that determines whether a major deviation alarm exists for the specified tagname.
<b>.MinorDevPct</b>	Read/write integer tagname .field used to monitors and/or controls the minor percent of deviation for alarm checking.
<b>.MinorDevStatus</b>	Read only discrete tagname .field used to determine whether a minor deviation alarm exists for the specified tagname.
<b>.Name</b>	Read/write message tagname .field used to display the actual name of the tagname. For example, it can be used to determine the name of an Alarm Group that a Group Variable is pointing to, or the name of a TagID tagname.

---

	It can also be written to in order to change the Alarm Group that a Group Variable is pointing to.
<b>.Normal</b>	Read only discrete tagname .field that is equal to 1 when there are no alarms for the specified name. This .field is valid for Alarm Groups and Group Variables as well as ordinary tagnames.
<b>.ROCPct</b>	Read/write .field used to monitor and/or control the rate of change for alarm checking.
<b>.ROCStatus</b>	Read only discrete .field used to determine whether a Rate-of-Change alarm exists for the specified tagname.



# Acknowledging Local Alarms

You can acknowledge local alarms by using the **.Ack (.field)** in an action or key script.

➤ **To create a local alarm acknowledge button:**

1. Create a 3-D button or any other object to which an action or key script can be linked.
2. Double-click the object or select it, and then on the **Special** menu, click **Animation Links**.
3. In the **Touch Pushbuttons** section of the animation link selection dialog box, click **Action**. The QuickScript editor will appear.
4. Type any of the following statements for the QuickScript:

**Ack \$System;** Acknowledges all local alarms in the system.

**Ack Group Name;** Acknowledges all local alarms in a specific Alarm Group.

**Ack Group Var;** Acknowledges all local alarms in a group indicated by the value of the Group Variable, an indirect Alarm Group tagname type.

**Ack Tagname;** Acknowledges a specific tagname's alarms.

or,

**\$System.Ack=1;** Acknowledges all local alarms in the system.

**Group Name.Ack=1;** Acknowledges all local alarms in a specific Alarm Group.

**Group Var.Ack=1;** Acknowledges all local alarms in a group indicated by the value of the Group Variable, an indirect Alarm Group tagname type.

**Tagname.Ack=1;** Acknowledges a specific tagname's alarms.

5. Click **OK**.

🔗 For more information on the QuickScript editor and its features, see Chapter 5 - Creating QuickScripts in OpenHMI.

## CHAPTER 7

# Trend Graphs

OpenHMI provides you with real-time trend display objects. You can configure trend objects to display graphical representations of multiple tagnames over time. Trends allow you to chart up to four pens (data values) and are created using special tools in WindowMaker. OpenHMI also provides you with complete control over the configuration of your trends. For example, you can specify the time span, value range, grid resolution, location of time stamps, location of value stamps, number of pens, and color attributes.

---


**Note:** OpenHMI runtime workstations do not support historical trends.

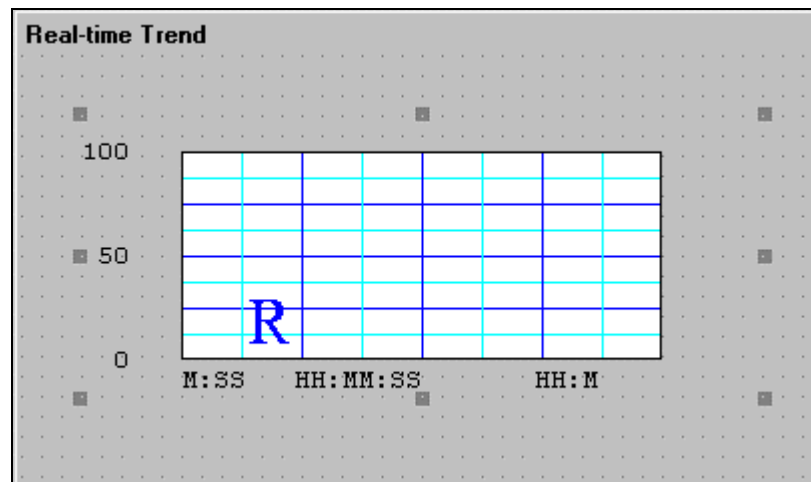
---

## Real-time Trends

Real-time trends are dynamic. They are updated continuously during runtime. They plot the changes of up to four local tagnames or expressions as they occur.

### Creating a Real-time Trend

-  **To create a real-time trend:**
  1. Select the real-time trend tool in the **Wizard Toolbar**.
  2. Click in the window, then drag the mouse diagonally to draw a rectangle the size that you want your trend to be. (You can draw the trend chart any size you choose, and there is no limit to the number of charts you can place on a screen.)
  3. Release the mouse. The real-time trend object appears in the window:



- ⌘ In runtime, the data is written in the trend from the right to the left.
- 4. Double-click the trend to open its configuration dialog box.
  - ⌘ A trend object is like any other object drawn in WindowMaker. It can be moved by grabbing it with the mouse or it can be resized by grabbing one of the object "handles." You can place multiple trends in a window.

### Configuring a Real-time Trend

The first time you paste a real-time trend object, the system default configuration settings are used. Once you have configured a real-time trend, the next one you create will, by default, be configured with the same settings.

- **To configure a real-time trend:**
  1. Double-click the trend or, with the trend selected, on the **Special** menu, click **Animation Links**. The **Real Time Trend Configuration** dialog box appears:

- ☞ If you right-click a text box in the real-time trend configuration dialog box, a menu will appear displaying the commands that you can apply to the selected text.

---

**Note** All entries made in the **Real Time Trend Configuration** dialog box are independent of the size of the trend and are not modifiable at runtime.

---

2. In the **Time Span** box, type length of time you want to display horizontally (x-axis) on the trend then select time increment option for the length of time.

For example, if you enter 30 for the **Time Span** then select **Min**, the horizontal time span of the chart will be 30 minutes long.

3. In the **Sample Interval** box, type the frequency at which the trend expression will be evaluated and the chart updated, then select the option for the time increment to which the number will relate.

For example, if you enter 10 for the **Interval** and select **Sec** for the time increment, the expression will be evaluated every 10 seconds.

4. In the **Color** group, click the **Chart Color** box to open the OpenHMI color palette. Click the color in the palette that you want to use for the trend's background.
5. In the **Color** group, click the **Border Color** box to open the OpenHMI color palette. Click the color in the palette that you want to use for the trend's border.

☞ Repeat this process for all color selections.

6. In the **Time Divisions** group, in the **Number of Major Div** box, type the number of major time divisions you want in the trend, and then select the color you want to use for the division lines.

---

**Note** The maximum time between major time divisions is (65536 sec) or 18 hours, 12 minutes, 16 seconds.

---

☞ The number of major time divisions must be an even multiple of the number of **Minor Div/Major Div**.

1. In the **Time Divisions** group, in the **Minor Div/Major Div** box, type the number of minor time divisions that you want to be visible within each major time division, and then select the color you want to use for the division lines.
2. In the **Time Divisions** group, select **Top Labels** if you want time labels displayed at the top of the trend.
3. In the **Time Divisions** group, select **Bottom Labels** if you want time labels displayed at the bottom of the trend.

☞ Your trend can have both top and bottom labels or no labels at all.
4. If you are using time labels, in the **Time Divisions** group, in the **Major Div/Time Label** box, type the number of time labels per major time division line that you want for the trend.
5. In the **Time Divisions** group, select the color you want to use for the major time division lines.
6. The settings in **Value Divisions** group are configured the same way as the settings in the **Time Divisions** group, except the minor and major value divisions set the vertical value (y-axis) range for the trend. This range uses Engineering Units and is the same for all trended tagnames.

☞ To display decimal points for the minor and major value divisions at runtime, they must be formatted here to do so. For example, 0.00 to 100.00.
7. In the **Expression** box, type the local tagname or expression that you want each **Pen** to trend.

☞ Up to four pens can be visible in a trend. The pens can be used to display any local tagname or an expression that contains one or more local tagnames. (Message type tags cannot be logged or trended.) The ability to trend expressions is useful in creating custom displays to show tagnames with widely different ranges.
8. Click the color box to select the color that you want each pen to use to plot each tagname in the trend.
9. In the **Width** box, type the number of pixels wide you want each pen to be.

☞ Selecting a pen width greater than 1 significantly reduces performance in trend updating and printing of the trend.
10. Click **Select Display Font** to access the **Font** dialog box to select the font, style and size that you want to use when you print the trend.
11. Select **Only update when in memory** if you want your trend to update only when it is displayed in the active window.

☞ If you do not select this option, the trend will always be updated, even if it is not in an open window. This may result in slightly slower system performance of the overall system.
12. Click **OK**.

➤ **To Increase real-time trending performance:**

1. Set the pen width to '1'.
2. Be sure no other objects are placed on top of the Real-time trend.

3. Lower the number of "samples" being taken.

For example, if you set the Time Span to 30 minutes and the Sample Interval to 2 seconds, the number of samples taken during the 30 minutes will be calculated as:

$$30 * 60 / 2 = 900$$

If you set the Time Span to 30 minutes and the Sample Interval to 5 seconds, the number of samples taken during the 30 minutes will be calculated as:

$$30 * 60 / 5 = 360$$

## CHAPTER 8

# I/O Communications

OpenHMI uses the Microsoft Dynamic Data Exchange (DDE), FastDDE, NetDDE and SuiteLink protocols to communicate with other Windows programs, I/O Servers and third-party I/O Server programs that are communicating with the real world.

## Supported Communication Protocols

Dynamic Data Exchange (DDE) is a communication protocol developed by Microsoft to allow applications in the Windows environment to send/receive data and instructions to/from each other. It implements a client-server relationship between two concurrently running applications. The *server* application provides the data and accepts requests from any other application interested in its data. Requesting applications are called *clients*. Some applications such as OpenHMI and Microsoft Excel can simultaneously be both a *client* and a *server*.

FastDDE provides a means of packing many proprietary Xycom Automation DDE messages into a single Microsoft DDE message. This packing improves efficiency and performance by reducing the total number of DDE transactions required between *client* and *server*. Although Xycom Automation's FastDDE has extended the usefulness of DDE for our industry, this extension is being pushed to its performance constraints in distributed environments.

NetDDE extends the standard Windows DDE functionality to include communication over local area networks and through serial ports. Network extensions are available to allow DDE links between applications running on different computers connected via networks or modems. For example, NetDDE supports DDE between applications running on IBM PCs connected via LAN or modem and DDE-aware applications running on non-PC based platforms under operating environments such as VMS and UNIX.

SuiteLink uses a TCP/IP based protocol. SuiteLink is designed specifically to meet industrial needs, such as data integrity, high-throughput, and easier diagnostics. This protocol standard is only supported on Microsoft Windows NT 4.0 or higher.

SuiteLink is not a replacement for DDE, FastDDE, or NetDDE. Each connection between a client and a server depends on your network situation. SuiteLink was designed specifically for high speed industrial applications and provides the following features:

- Value Time Quality (VTQ) places a time stamp and quality indicator on all data values delivered to VTQ-aware clients.
- Extensive diagnostics of the data throughput, the server loading, computer resource consumption, and network transport are made accessible through the Microsoft Windows NT operating system performance monitor. This feature is critical for the scheme and maintenance of distributed industrial networks.
- Consistent high data volumes can be maintained between applications regardless if the applications are on a single node or distributed over a large node count.
- The network transport protocol is TCP/IP using Microsoft's standard Winsock interface.

## Configuring DDE Shares

OpenHMI is shipped with the NetDDE product, portions of which were licensed to Microsoft as *Network DDE* for use in Windows 95, and Windows NT. The NetDDE provided with OpenHMI conveys additional capabilities over the Microsoft Network DDE. NetDDE provided with OpenHMI is used only on the Windows 95 operating system while, Xycom Automation's *NetDDE Extensions* is used with the Windows NT operating system for configuring WinSock for use with Microsoft Network DDE.



If you are using the Windows NT operating system, you will need to set up a DDE share for any node containing I/O resources that other View nodes may need to access. For example, if you have a node running GE Genius I/O Server, you need to create a share for that I/O Server.

## Windows NT Operating System

Microsoft includes *NT Network DDE* with its Windows NT operating system. In order to allow *NT Network DDE* to act as a resource for OpenHMI, DDE shares must be created for all nodes running on the Windows NT operating system with I/O resources that OpenHMI View nodes may need. (The "Server" referred to here is synonymous with Data Source it is not to be confused with the NT Server operating system.)

## SuiteLink Communication Protocol

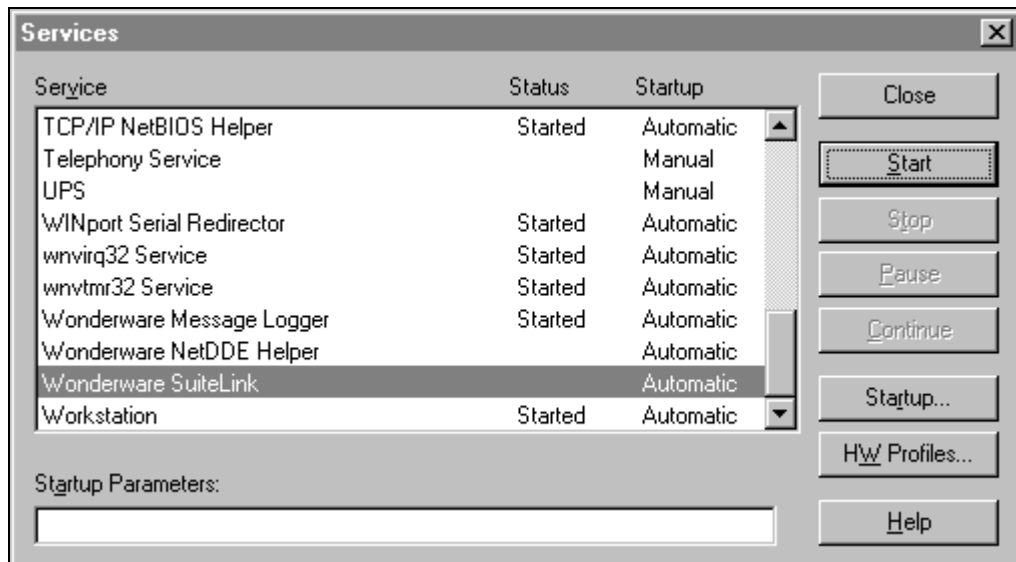
OpenHMI is shipped with the communications protocol SuiteLink. Wonderware SuiteLink uses a TCP/IP based protocol. SuiteLink is designed specifically to meet industrial needs, such as data integrity, high-throughput, and easier diagnostics. This protocol standard is only supported on Microsoft Windows NT 4.0 or higher.

## SuiteLink

SuiteLink uses a TCP/IP based protocol. SuiteLink is designed specifically to meet industrial needs, such as data integrity, high-throughput, and easier diagnostics. This protocol standard is only supported on Microsoft Windows NT 4.0 or higher.

➤ **To use the SuiteLink communication protocol:**

1. You must have Microsoft TCP/IP configured and working properly.
  - ☞ Refer to your Windows NT operating system's Help for complete details on installing and configuring Microsoft TCP/IP.
2. SuiteLink must be running as a service. If for some reason SuiteLink has been stopped, you will need to start it again.
  - ☞ SuiteLink is automatically installed when you install OpenHMI and by default, it is configured to startup automatically as an NT Service.
    - a) To start SuiteLink as an NT Service, open the Windows Control Panel.
    - b) Double-click **Services**. The **Services** dialog box appears:



- c) Select **Wonderware SuiteLink**, and then click **Start**.
- d) Click **Close**.

## The OpenHMI I/O Address Convention

OpenHMI identifies an element of data in an I/O Server program by using a three-part naming convention that includes the *application name*, *topic name* and *item name*. To obtain data from another application the *client* program (OpenHMI) opens a channel to the *server* program by specifying these three items.

In order for OpenHMI to acquire a data value from another application, it must also know the name of the *application* providing the data value, the name of the *topic* within the application that contains the data value, and the name of the specific *item* within the *topic*. In addition, OpenHMI needs to know the data's type; discrete, integer, real (floating point), or message (string). This information determines the I/O type for the tagname when it is defined in the OpenHMI database. Now, when WindowViewer is running, it will automatically perform all of the actions required to acquire and maintain the value of this *item*.

For example, in the case of Excel, the *application name* is "Excel," the *topic name* is the name of the specific spreadsheet that contains the data and the *item name* is the identification of the cell on the spreadsheet to/from which the data is to be read/written.

## The OpenHMI I/O Address

When another Windows application requests a data value from OpenHMI, it also must know the three I/O address items. The following describes the I/O address convention for OpenHMI:

1. **VIEW** (*application name*) identifies the OpenHMI runtime program that contains the data element.
2. **TAGNAME** (*topic name*) is the word always used when reading/writing to a tagname in the OpenHMI database.
3. **ActualTagname** (*item name*) is the actual tagname defined for the item in the OpenHMI Tagname Dictionary.

For example, to access a data value in OpenHMI from Excel, a DDE Remote Reference formula would be entered in the cell into which the data value is to be written:

**=VIEW|TAGNAME!'ActualTagname'**

---

**Note** If you are networking using NetDDE, the *application name* portion of the I/O address must be prefixed with the remote node's name preceded by two backslashes and followed by one backslash. For example:

**\\NodeName\VIEW|TAGNAME!'ActualTagname'**

---

# OpenHMI Access Names

When you create I/O type tagnames, they must be associated with an Access Name. Access Names contain the information that is used to communicate with other I/O data sources including the node name, application name and topic name.

➤ **To create an Access Name:**

1. On the **Special** menu, click **Access Names**, or in the Application Explorer under **Configure**, double-click **Access Names**. The **Access Names** dialog box appears:

☞ In the Application Explorer, you can right-click **Access Names**, and then click **Open**. You can also create Access Names while you are defining an I/O type tagname in the Tagname dictionary.



2. Click **Add**. The **Add Access Name** dialog box appears:

3. In the **Access Name** box type the name you want OpenHMI to use to this Access Name. (For simplicity, use the same name that you will use for the *topic name* here.)

☞ OpenHMI uses Access Names to reference real-time I/O data. Each Access Name equates to an I/O address, which can contain a Node, Application,

and Topic. In a distributed application, I/O references can be set up as global addresses to a network I/O Server or local addresses to a local I/O Server.

4. If the data resides in a network I/O Server, in the **Node Name** box, type the remote node's name.
5. In the **Application Name** box, type the actual program name for the I/O Server program from which the data value will be acquired. In this case the value is coming from the Modbus I/O Server, therefore **MODBUS** is used. **Do not** enter the **.exe** extension portion of the program name.
6. In the **Topic Name** box, type the *topic name* you want to access.
  - ☞ The **Topic Name** is an application-specific sub-group of data elements. In the case of data coming from a I/O Server program, the *topic name* is the **exact** same name configured for the *topic* in the I/O Server program. When communicating with Microsoft Excel, the *topic name* must be the name given to the spreadsheet when it was saved. For example, Book1.xls.
7. Select the protocol that you are using.
8. Select **Advise all items** if you want the server program to poll for all data whether or not it is in visible windows, alarmed, logged, trended or used in a script. Selecting this option will impact performance, therefore its use is not recommended.
9. Select **Advise only active items** if you want the server program to poll only points in visible windows and points that are alarmed, logged, trended or used in any script.
  - ☞ A touch pushbutton action script will not be polled unless it appears in a visible window.
10. Click **OK** to accept the new Access Name and close the dialog box. The **Access Names** dialog box will reappear displaying the new Access Name selected in the list:



11. Click **Close** to close the dialog box and return to your tagname definition.
- **To modify or delete an Access Name:**
12. On the **Special** menu, click **Access Names**, or in the Application Explorer under **Configure**, double-click **Access Names**. The **Access Names** dialog box appears:
    - ☞ In the Application Explorer, you can right-click **Access Names**, and then click **Open**.



13. To change an Access Name's definition, select it in the list, and then click **Modify**. The **Modify Access Name** dialog box will appear. Make your required changes, and then click **OK**. The **Access Names** dialog box reappears. Click **Close** or repeat this procedure if you need to modify other defined Access Names.
14. To delete an Access Name, select it in the list, and then click **Delete**. A message box will appear asking you to confirm the deletion of the selected Access Name. Click **Yes** to delete it or click **No** to cancel the deletion. Click **Close** or repeat this procedure if you need to delete other defined Access Names.

---

**Note** Access Names that are used in tagnames cannot be deleted.

---

## Defining an I/O Item in OpenHMI

OpenHMI can receive data from other local or remote Windows applications when you define I/O type tagnames in the Tagname Dictionary. Each I/O type tagname references a valid *item* in the I/O Server program.

☞ For more information on distributed applications see, Chapter 3 - Creating a Distributed Application.

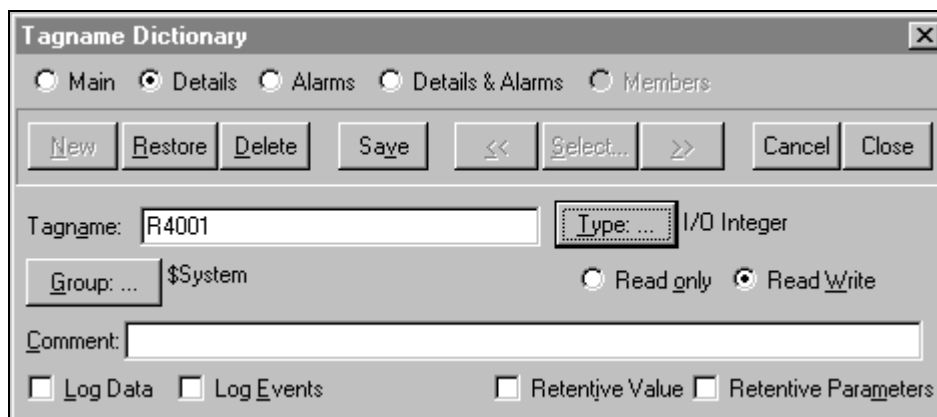
### ➤ To define an I/O type tagname:

1. On the **Special** menu, click **Tagname Dictionary**, or in the Application Explorer double-click **Tagname Dictionary**. The **Tagname Dictionary** dialog box appears.

☞ For more information on defining I/O tagnames, see Chapter 3 - Tagname Dictionary

2. Click **New**. The **Tagname** box clears.

☞ If you right-click any of the text entry boxes in any of the Tagname Dictionary dialog boxes, a menu will appear displaying the commands that you can apply to the selected text.



☞ The first time you access the Tagname Dictionary, the definition for the internal system tagname **\$AccessLevel** is displayed. Once you define tagnames in the Tagname Dictionary, when you access it again, the last edited tagname's definition is displayed.

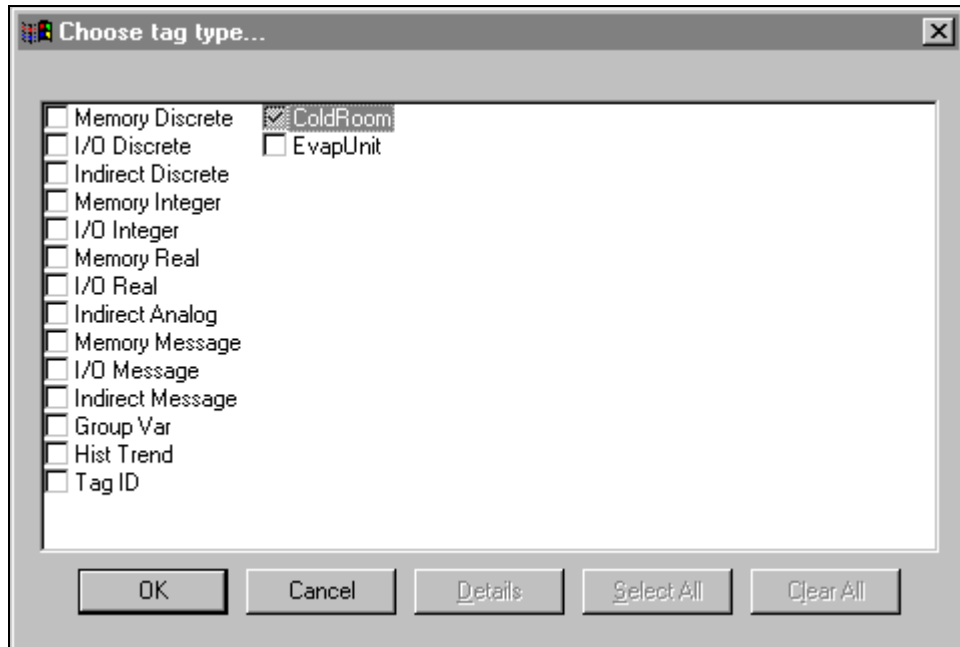
3. In the **Tagname** box, type the name you want to use for the new tagname.

☞ Tagnames can be up to 32 characters long and must begin with an alpha character (A-Z or a-z). The remaining characters can be A-Z, a-z, 0-9, !, @, -, ?, #, \$, %, \_ \ and &.

Tagnames are also auto-indexed. For example, if you enter and save tagname R4001, and then click **New**, the tagname will automatically be indexed to R4002. If a tagname contains a character separating numbers, it is auto-indexed by the first whole number OpenHMI finds. For example, N7-0 would be indexed as N7-1. Positive changes only are permitted. For example, R4002 to R4003, R4003 to R4004 and so on.

4. Click **Type**. The **Choose tagname type** dialog box appears.





5. Select the I/O type for the tagname as follows:

**I/O Discrete**     input/output value of either True (1) or False (0)  
**I/O Integer**     (whole number) input/output value  
**I/O Real**         floating (decimal) point input/output value  
**I/O Message**    (string) input/output value

6. Once you select the I/O type, and then click **OK**. The respective "details" dialog box for the selected I/O type will appear. For example, if you select I/O Integer, the following dialog box appears:

☞ If the "Details" dialog box does not appear, click **Details** at the top of the screen.

7. Specify all the required details for defining the *item*.
8. Click **Access Name**. The **Access Names** dialog box appears:



9. Double-click the Access Name that you want to use in the list or select it, and then click **Close**.
10. The Access Name that you selected (now associated with this tagname definition) appears adjacent to the **Access Name** button in the details dialog box. For example:

11. In the **Item** box, type the *item name* for the data value in the I/O Server program.

---

**Note** It is important to understand that the "tagname" is the name used within OpenHMI to refer to a data value. The **Item** is the name used by a remote Windows application to refer to the same value. These names do not have to be the same but, it is recommended when applicable to use the same names. Also, if the **Item** is a cell in Excel, it must be specified either by its defined name in Excel, or by its row/column identification. For example, R1C1.

---

12. Click **Close**.

# Monitoring the Status of an I/O Conversation

WindowViewer supports a built-in *topic name* called **IOStatus** (**DDEStatus** in versions prior to OpenHMI 7.0) that can be used to monitor the status of specific I/O conversations.

## Using IOStatus Topic Name

Let's assume that WindowViewer (View) is communicating with the Simulate I/O Server to a PLC that has been defined in the I/O Server with **PLC1** for its *topic name*.

☞ (Simulate is a generic I/O Server that is intended to be used as a training tool. Simulate is included with OpenHMI.)

### ➤ To monitor the status of I/O communications:

1. On the **Special** menu, click **Tagname Dictionary**, or in the Application Explorer, double-click **Tagname Dictionary**. The **Tagname Dictionary** dialog box appears.
2. Create an **I/O Discrete** type tagname. (In this example, for simplicity, we will make our tagname the same as the *topic name* that we want to monitor):

☞ When you are monitoring a I/O conversation using **IOStatus**, you must define at least one I/O type tagname to the Access Name being monitored.

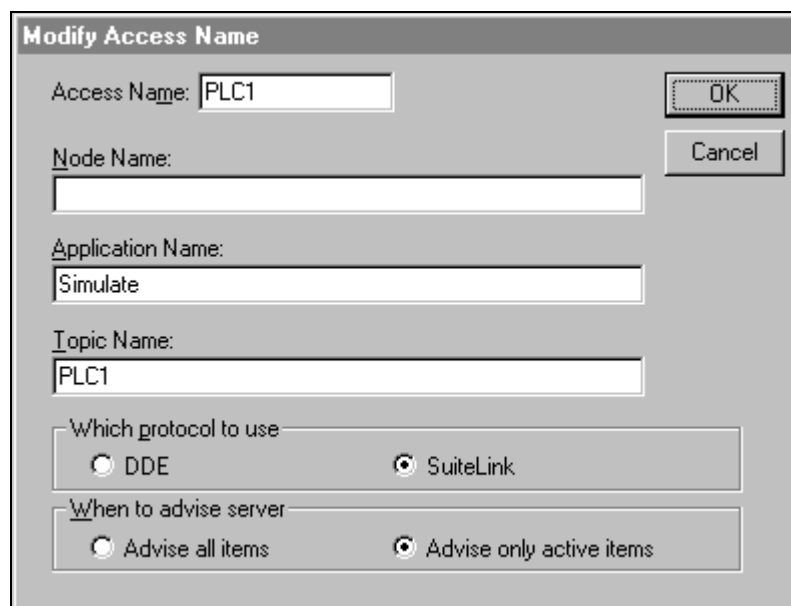
The screenshot shows the 'Tagname Dictionary' dialog box with the following details:

- Tags:** Main (selected), Details, Alarms, Details & Alarms, Members
- Buttons:** New, Restore, Delete, Save, <<, Select..., >>, Cancel, Close
- Tagname:** PLC1
- Type:** I/O Discrete
- Group:** \$System
- Permissions:** Read only (unselected), Read Write (selected)
- Comment:** (empty text box)
- Options:** Log Data (unselected), Log Events (unselected), Retentive Value (unselected)
- Initial Value:** On (unselected), Off (selected)
- Input Conversion:** Direct (selected), Reverse (unselected)
- On Msg:** (empty text box)
- Off Msg:** (empty text box)
- Access Name:** Unassigned
- Item:** (empty text box)
- Use Tagname as Item Name:** (unselected checkbox)

3. Click **Access Name** to assign the tagname to an Access Name definition that defines **IOStatus** for its *topic name*. The **Access Name Definition** dialog box appears:



- ☞ Notice that an Access Name definition called **PLC1** (the topic we want to monitor) currently exists. To be sure that this is the correct Access Name (its **Topic Name** is **PLC1**), click **Modify** to view the definition:



- ☞ Finding the Access Name containing the right *topic name* in this example was easy because we kept the tagname and the **Topic Name** the same.
4. Click **Cancel** to close the dialog box and return to the initial **Access Name Definition** dialog box.
  5. Click **Add**. The **Add Access Name** dialog box will appear:

**Add Access Name**

Access Name:

Node Name:

Application Name:

Topic Name:

Which protocol to use

DDE  SuiteLink

When to advise server

Advise all items  Advise only active items

6. In the **Access Name** box, type **IOStatus**.
7. Since you are going to monitor the status in WindowViewer, in the **Application Name** box, type "View."
8. In the **Topic Name** box, type the OpenHMI internal *topic*, **IOStatus**.
9. Select **Advise only active items**.
10. Click **OK** to close the dialog box. The initial **Access Name Definition** dialog box reappears displaying your new **Access Name**, **IOStatus**, in the list:

**Access Names**

11. Click **Close** to close the dialog box and associate the new **Access Name** with your **I/O Discrete** tagname:

Initial Value:  On  Off

Input Conversion:  Direct  Reverse

On Msg:

Off Msg:

Access Name:

Item:

Use Tagname as Item Name

12. In the **Item** box, type the actual **Topic Name** that you want to monitor. In this case, **PLC1**.

☞ Since your tagname is the same as the **Topic Name**, you can select **Use Tagname as Item Name** and automatically enter it for the **Item**.

---

**Note** When using the built-in topic **IOStatus (DDEStatus** prior to OpenHMI Version 7.0) to monitor an I/O conversation, the name you type in the **Access Name** box is always also used for the **Item**.

---

## Using IOStatus Topic Name in Excel

Excel can also be used to perform this same type of monitoring by entering the same information in a formula in a spreadsheet cell. For example, to monitor the same topic as above, the following would be entered:

`=view|IOStatus!'PLC1'`

# Monitoring I/O Server Communications Status

For each *topic name* being used, there is a built-in discrete *item*, **Status**, that you can use to monitor the state of your communications with the I/O Servers program.

**Status** is set to "0" when communications with the device fails (cable disconnected, PLC is powered down, and so on.) and set to "1" when communications is successful.

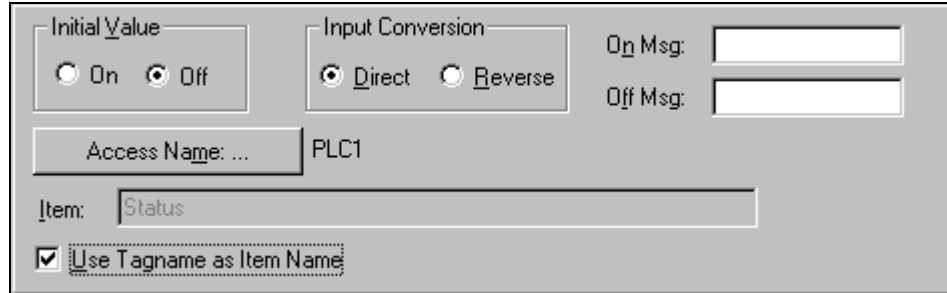
---

**Note** When you monitor the status of a topic using the **Status** item, there must at least one I/O point active to the topic being monitored.

---

From OpenHMI, you can read the state of the server communications by defining a tagname and associating it with the *topic* configured for the device by using the word **Status** as the *item name*. For example, if WindowViewer is communicating with a PLC using the Simulate I/O Server, the Access Name definition would be:

To monitor the status of all communication to the *topic*, PLC1, you would create the following tagname definition:



The image shows a configuration dialog box for PLC communication. It has a grey background and a white border. The dialog is divided into several sections:

- Initial Value:** A group box containing two radio buttons: "On" (unselected) and "Off" (selected).
- Input Conversion:** A group box containing two radio buttons: "Direct" (selected) and "Reverse" (unselected).
- On Msg:** A text input field.
- Off Msg:** A text input field.
- Access Name:** A text input field containing "PLC1".
- Item:** A text input field containing "Status".
- Use Tagname as Item Name:** A checked checkbox.

From Excel, you can read the status of the PLC communications by entering the following formula in a cell:

**=SIMULATE|PLC1!'STATUS'**



# Monitoring Multiple Input Device Status

This section describes how you can display the status of an object using multiple inputs.

## Example 1

In this example, the status of a spring-return motorized valve using two inputs is being viewed. The two inputs represent a pair of limit switches installed on the valve. One input is only on when the valve is in the open position and off when the valve is in travel or closed. The other input is only on when the valve is in the closed position and off when the valve is in the travel or open position. A truth table of the inputs would appear as follows:

Valve Truth Table 1			
Input #1 (opened)	Input #2 (closed)	Valve Position	Result
1	0	Opened	$1 + 0 = 1$
0	1	Closed	$0 + 1 = 1$
0	0	InTravel	$0 + 0 = 0$
1	1	InValid Position	$1 + 1 = 1$
0 = OFF    1 = ON			

The inputs can be weighed by multiplying the closed input by 2. The results of the valve positions then change to the following:

Valve Truth Table 2 (Input #2 x2)			
Input #1 (opened)	Input #2 (closed)	Valve Position	Result
1	$0 \times 2 = 0$	Opened	$1 + 0 = 1$
0	$1 \times 2 = 2$	Closed	$0 + 2 = 2$
0	$0 \times 2 = 0$	InTravel	$0 + 0 = 0$
1	$1 \times 2 = 2$	InValid Position	$1 + 2 = 3$
0 = OFF    1 = ON			

---

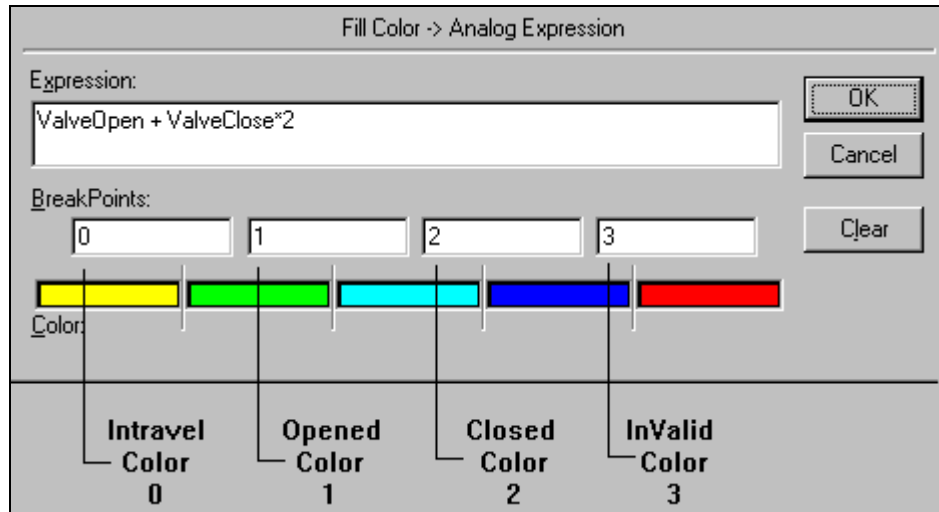
**Note** The invalid position can be used to show a defective limit switch.

---

Now that there is a significant numerical difference between the valve positions, a **Fill Color - Analog** animation link can be used to display the valve status.

☞ For more information on creating animation links, see Chapter 4 - Creating Animation Links.

Two I/O Discrete tagnames are created. One for the valve open input and one for the valve closed input. For example, **ValveOpen** and **ValveClosed**. An object is created to display the valve status. This object is assigned to a **Fill Color - Analog** animation link with the following properties:



**Example 2**

In this example, one more input has been added to the existing two. This new input is the actual output to the valve that causes it to open. The new input will be on when the valve is opening or open, and off when the valve is closing or closed. The new truth table appears as follows:

Input #1 (opened)	Input #2 (closed)	Input #3 (open)	Valve Position	Result
0	0	1	Opening	$0 + 0 + 1 = 1$
1	0	1	Opened	$1 + 0 + 1 = 2$
0	0	0	Closing	$0 + 0 + 0 = 0$
0	1	0	Closed	$0 + 1 + 0 = 1$

0 = OFF    1 = ON

Once again the inputs are weighed. As previously explained, the closed input will be multiplied by 2 and the new input will be multiplied by 4 for the following results:

Input #1 (opened)	Input #2 (closed)	Input #3 (open)	Valve Position	Result
0	$0 \times 2 = 0$	$1 \times 4 = 4$	Opening	$0 + 0 + 4 = 4$
1	$0 \times 2 = 0$	$1 \times 4 = 4$	Opened	$1 + 0 + 4 = 5$
0	$0 \times 2 = 0$	$0 \times 4 = 0$	Closing	$0 + 0 + 0 = 0$
0	$1 \times 2 = 2$	$0 \times 4 = 0$	Closed	$0 + 2 + 0 = 2$

0 = OFF    1 = ON

Another I/O Discrete tagname (**Valve**) is created for the new open input and assigned to a **Fill Color - Analog** animation link with the following properties:

Fill Color -> Analog Expression

Expression:  
ValveOpen + ValveClosed\*2 + Valve\*4

BreakPoints:  
0 2 4 5

Color:

Closing Color 0    Closed Color 2    Opening Color 4    Opened Color 5

OK  
Cancel  
Clear

Using this method, additional inputs can be used. The fourth input would be multiplied by 8, the fifth by 16, and so on.vc

# Index

## \$

\$AccessLevel, 2-81, 2-87, 2-91, 2-93, 3-8, 3-40  
 \$AlarmPrinterError, 3-40  
 \$AlarmPrinterNoPaper, 3-40  
 \$AlarmPrinterOffline, 3-40  
 \$AlarmPrinterOverflow, 3-40  
 \$ApplicationChanged, 3-40  
 \$ApplicationVersion, 3-40  
 \$ChangePassword, 2-91, 3-40  
 \$ConfigureUsers, 2-91, 3-40  
 \$Date, 3-41  
 \$DateString, 3-41  
 \$DateTime, 3-41  
 \$Day, 3-41  
 \$Hour, 3-41  
 \$InactivityTimeout, 2-81, 2-93  
 \$InactivityWarning, 2-81, 2-92, 2-93, 3-41  
 \$LogicRunning, 2-84  
 \$Minute, 3-41  
 \$Month, 3-41  
 \$Msec, 3-41  
 \$NewAlarm, 3-41  
 \$ObjHor, 3-41  
 \$ObjVer, 3-41  
 \$Operator, 2-87, 2-91  
 \$OperatorEntered, 2-91, 3-42  
 \$PasswordEntered, 2-91, 3-42  
 \$Second, 3-42  
 \$StartDdeConversations, 3-42  
 \$System, 3-7, 3-42  
 \$Time, 3-42  
 \$TimeString, 3-42  
 \$Year, 3-42

## %

% Deviation, 3-16

.

.Ack, 3-44, 6-20  
 .Alarm, 3-44, 6-20  
 .AlarmDevDeadband, 3-44, 6-20  
 .AlarmEnable, 6-20  
 .AlarmEnabled, 3-44  
 .AlarmValDeadband, 3-44, 6-20  
 .Comment, 3-45  
 .DevTarget, 3-45, 6-20  
 .HiHiLimit, 3-45, 6-20  
 .HiHiStatus, 3-45, 6-20  
 .HiLimit, 3-45, 6-20  
 .HiStatus, 3-45, 6-20  
 .LoLimit, 3-45, 6-20  
 .LoLoLimit, 3-45, 6-20  
 .LoLoStatus, 3-45, 6-20  
 .LoStatus, 3-45, 6-20

.MajorDevPct, 3-45, 6-20  
 .MajorDevStatus, 3-45, 6-20  
 .MaxEU, 3-10, 3-45  
 .MaxRange, 3-46  
 .MaxRaw, 3-39, 3-46  
 .MinEU, 3-10, 3-45  
 .MinorDevPct, 3-46, 6-20  
 .MinorDevStatus, 3-46, 6-20  
 .MinRange, 3-46  
 .MinRaw, 3-39, 3-46  
 .Name, 3-46, 6-21  
 .Normal, 3-46, 6-21  
 .OffMsg, 3-45, 3-46  
 .OnMsg, 3-45, 3-46  
 .Quality, 3-47  
 .QualityLimit, 3-47  
 .QualityLimitString, 3-47  
 .QualityStatus, 3-47  
 .QualityStatusString, 3-47  
 .QualitySubstatus, 3-47  
 .QualitySubstatusString, 3-47  
 .RawValue, 3-47  
 .Reference, 3-47  
 .ReferenceComplete, 3-47  
 .ROCpct, 3-47, 6-21  
 .ROCStatus, 3-47, 6-21  
 .TimeDate, 3-48  
 .TimeDateString, 3-48  
 .TimeDateTime, 3-48  
 .TimeDay, 3-48  
 .TimeHour, 3-48  
 .TimeMinute, 3-49  
 .TimeMonth, 3-49  
 .TimeMsec, 3-49  
 .TimeSecond, 3-49  
 .TimeTime, 3-49  
 .TimeTimeString, 3-49  
 .TimeYear, 3-49  
 .Unack, 3-44, 3-49  
 .UpdateCount, 3-49  
 .Value, 3-49

## A

About dialog box, 1-34  
 Abs(), 5-39  
 Access Names
 

- Advise all Items, 8-8
- Advise only Active Items, 8-8
- Application Name, 8-8
- Creating, 8-7
- Deleting, 8-8
- Item Names, 8-12
- Modifying, 8-8
- Topic Name, 8-8

 Access Names dialog box, 8-7  
 Ack, 5-41  
 Acknowledge Alarms, 6-22  
 Action Touch Pushbutton Links, 4-12, 4-14

- Action Touch Pushbutton Scripts, 5-2
- ActivateApp(), 5-40
- ActiveX Control Browser dialog box, 5-20
- ActiveX Control Installation dialog box, 2-65
- ActiveX Control Properties dialog box, 2-64
- ActiveX Controls, 2-4
  - Accessing methods/properties, 2-72
  - Adding to the Toolbar, 2-66
  - Associating tagnames to properties, 2-69
  - Association direction, 2-70
  - Association symbols, 2-70
  - Configuring Properties, 2-69
  - Configuring the Control, 2-68
  - Container, 2-64
  - Control Browser dialog box, 2-72, 2-76
  - Control Name, 2-68, 2-73, 2-76
  - Editing, 2-65
  - Event parameters, 2-74
  - Event Scripts, 2-77, 5-2, 5-18
    - Importing, 5-36
  - Events, 2-64
  - Importing Event Scripts, 2-79
  - Installing, 2-65
  - Methods, 2-64, 2-70
  - Not Supported by InTouch, 2-65
  - Pasting in a window, 2-66
  - Properties dialog box, 2-65
  - Removing from an application, 2-65
  - Removing from the toolbar, 2-67
  - Reusing Event Scripts, 2-77
  - Tagname/Property association, 2-70
  - This Control, 2-73
  - This Event, 2-76
  - Using in InTouch, 2-64
- ActiveX Event Scripts dialog box, 5-19
- Add a Color dialog box, 1-22
- Add Access Names dialog box, 8-7
- Adding
  - ActiveX controls to the toolbar, 2-66
  - Applications to the Application Explorer, 1-8
  - Wizards to the toolbar, 2-54
- Addition ( + ) Operator, 5-30
- Add-on Program Script Functions, 5-5
- Adjusting a color's attributes, 1-23
- Alarm .Fields
  - .Ack, 3-44, 6-20
  - .Alarm, 3-44, 6-20
  - .AlarmDevDeadband, 3-44, 6-20
  - .AlarmEnable, 6-20
  - .AlarmEnabled, 3-44
  - .AlarmValDeadband, 3-44, 6-20
  - .DevTarget, 3-45, 6-20
  - .HiHiLimit, 3-45, 6-20
  - .HiHiStatus, 3-45, 6-20
  - .HiLimit, 3-45, 6-20
  - .HiStatus, 3-45, 6-20
  - .LoLimit, 3-45, 6-20
  - .LoLoLimit, 3-45, 6-20
  - .LoLoStatus, 3-45, 6-20
  - .LoStatus, 3-45, 6-20
  - .MajorDevPct, 3-45, 6-20
  - .MajorDevStatus, 3-45, 6-20
  - .MinorDevPct, 3-46, 6-20
  - .MinorDevStatus, 3-46, 6-20
  - .Name, 6-21
  - .Normal, 6-21
  - .ROCPct, 3-47, 6-21
  - .ROCStatus, 3-47, 6-21
  - .Unack, 3-44, 3-49
- Alarm Configuration dialog box, 6-11
- Alarm Printing Properties dialog box, 6-17
- Alarm Properties dialog box, 6-16
- Alarm Server Node, 2-82
- Alarms
  - Acknowledging, 6-22
  - Alarm .Fields, 6-20
  - Alarm Group Hierarchy, 6-4
  - Alarm Groups
    - Creating, 6-4
    - Deleting, 6-6
    - Modifying, 6-6
  - Alarm Types
    - Alarm Value, 6-2
    - Deviation, 6-2
    - Rate-Of-Change, 6-2
  - Alarm/Events Printing, 6-17
- Analog
  - % Deviation, 3-16
  - Alarm Value, 3-15
  - Defining analog alarm conditions, 3-15
  - Deviation Deadband %, 3-16
  - Major Deviation, 3-16
  - Minor Deviation, 3-16
  - Rate of Change, 3-16
  - Target, 3-16
  - Types, 3-15
  - Value Deadband, 3-16
- Configuring a Distributed Alarm/Event Object, 6-16
- Configuring Standard Alarm Object, 6-11
- Creating Standard Alarm Object, 6-10
- Discrete
  - Alarm State, 3-15
  - Defining discrete alarm conditions, 3-15
- Event Types
  - ACK, 6-2
  - ALM, 6-2
  - DDE, 6-2
  - EVT, 6-2
  - LGC, 6-2
  - OPR, 6-2
  - RTN, 6-2
  - SYS, 6-2
  - USER, 6-2
- Formatting Alarm/Event Message, 6-13
- Groups, 3-7
- History Objects, 6-9
- Next Page Button, 6-12
- Previous Page Button, 6-12
- Printing, 6-18
- Priorities, 6-3
- Priority, 3-8, 3-15, 3-16
- Summary Objects, 6-9
- Aligning Objects, 2-29

- Bottom, 1-17
- Bottom Edges, 2-33
- Centerpoints, 1-17, 2-33
- Centers, 1-17, 2-30
- Left Edges, 1-17, 2-29
- Middles, 1-17, 2-32
- Right Edges, 1-17, 2-31
- Top, 1-17
- Top Edges, 2-31
- Alignment Tools
  - Bottom Edges, 1-17, 2-33
  - Centerpoints, 1-17, 2-33
  - Centers, 1-17, 2-30
  - Left Edges, 1-17, 2-29
  - Middles, 1-17, 2-32
  - Right Edges, 1-17, 2-31
  - Top Edges, 1-17, 2-31
- Allow CTRL-Break to stop scripts, 2-84, 2-85
- Allow WindowViewer to dynamically change resolution, 2-95
- Always Load Windows from Disk, 2-81
- Analog Input Links, 4-9
- Analog Value Display Links, 4-30
- AND Operator, 5-31
- Animating Objects, 4-6
- Animation Links
  - Accessing Tagname .Fields, 4-5
  - Accessing the Tag Browser, 4-4
  - Apply Color Links, 4-4
  - Assigning Key Equivalents, 4-3
  - Display Links, 4-17
    - Color Links
      - Analog, 4-17
      - Analog Alarm, 4-17, 4-20
      - Analog Color Fill, 4-18
      - Discrete, 4-17
      - Discrete Alarm, 4-17, 4-19
      - Discrete Color Fill, 4-17
    - Fill Color, 4-17, 4-18, 4-19, 4-20
    - Line Color, 4-17, 4-18, 4-19, 4-20
    - Location, 4-17, 4-22
    - Miscellaneous, 4-17
      - Blink, 4-25
      - Disable, 4-25, 4-28
      - Orientation, 4-25, 4-26
      - Visibility, 4-25
    - Object Size, 4-17, 4-21
    - Percent Fill, 4-17, 4-23
    - Text Color, 4-17, 4-18, 4-19, 4-20
    - Value Display, 4-17
      - Analog, 4-29, 4-30
      - Discrete, 4-29
      - String, 4-29, 4-30
- Features, 4-2
- Object Type dialog box, 4-2
- Selection dialog box, 4-2
- Touch Links, 4-8
  - Sliders, 4-8
  - Touch Pushbuttons, 4-8
  - User Inputs, 4-8
- Touch Pushbutton Touch Links
  - Action, 4-12, 4-14
  - Discrete Value, 4-12, 4-13
  - Hide Window, 4-13, 4-16
  - Show Window, 4-13, 4-16
- User Input Touch Links
  - Analog, 4-9
  - Discrete, 4-9
  - Slider, 4-11
  - String, 4-10
- App Development dialog box, 2-97
- Application Action Script Dialog Box, 5-9
- Application Directory to Dump From dialog box, 3-54
- Application Explorer, 1-1, 1-2, 1-4
  - Adding Applications, 1-8
  - Add-on Programs, 1-7
  - Applications, 1-8
  - Collapsing, 1-5, 1-6
  - Configure, 1-7
  - Cross Reference, 1-7
  - Docking, 1-4
  - Expanding, 1-5, 1-6
  - Floating, 1-4
  - Hiding, 1-5
  - Navigating, 1-5
  - Right-click menu, 1-4
  - Scripts, 1-6
  - Showing, 1-5
  - Tagname Dictionary, 1-7
  - Windows, 1-6
- Application Explorer Tool, 1-16
- Application Name, 8-5
- Application Scripts, 5-2
  - On Shutdown, 5-9
  - On Startup, 5-9
  - While Running, 5-9
- Application Scripts Editor dialog box, 5-9
- Applying color to objects, 1-21
- Applying security to your application, 2-87
- ArcCos(), 5-39
- ArcSin(), 5-39
- ArcTan(), 5-39
- Arrange Toolbar, 1-11, 1-17, 2-18, 2-29
  - Align Bottom Tool, 1-17, 2-33
  - Align Center Tool, 1-17, 2-30
  - Align Centerpoints Tool, 1-17, 2-33
  - Align Left Tool, 1-17, 2-29
  - Align Middle Tool, 2-32
  - Align Middles Tool, 1-17
  - Align Right Tool, 1-17, 2-31
  - Align Top Tool, 1-17, 2-31
  - Break Cell Tool, 1-18, 2-38
  - Break Symbol Tool, 1-18, 2-38
  - Bring to Front Tool, 1-17, 2-34
  - Flip Horizontal Tool, 1-18, 2-36
  - Flip Vertical Tool, 1-18, 2-36
  - Make Cell Tool, 1-18, 2-38
  - Make Symbol Tool, 1-18, 2-38
  - Reshape Object Tool, 1-18, 2-26
  - Rotate Clockwise Tool, 1-18, 2-36
  - Rotate CounterClockwise Tool, 1-18, 2-36
  - Send to Back Tool, 1-17, 2-34
  - Space Horizontal Tool, 1-18, 2-35
  - Space Vertical Tool, 1-18, 2-35

Arranging Objects  
 Bring to Front, 1-17, 2-34  
 In a Window, 2-29  
 Send to Back, 1-17, 2-34  
 Spacing Horizontally, 1-18, 2-35  
 Spacing Vertically, 1-18, 2-35  
 Assignment ( = ) Operator, 5-31  
 Attributes  
 Dynamic, 2-2  
 Static, 2-2  
 Automatically Changing the System Time, 2-102  
 Automatically Logging Off the System, 2-91

## B

Beep when objects touched, 2-82  
 Bit Fields, 3-49  
 Bitmap Objects, 2-3  
 Bitmaps, 2-40  
 Container, 2-43  
 Importing, 2-40  
 Making original size, 2-41  
 Pasting, 2-42  
 Pasting from Clipboard, 2-42  
 Rotating, 2-40  
 Tool, 1-16, 2-40, 2-42, 2-43  
 Transparent, 1-24, 2-40, 2-43  
 Transparent Color Tool, 1-15  
 Bitwise AND ( & ) Operator, 5-31  
 Blink Links, 4-25  
 Blink object speeds, 2-82  
 Blotter Tool, 1-24, 2-44  
 Bold Tool, 1-14  
 Break Cell Tool, 1-18, 2-38  
 Break Symbol Tool, 1-18, 2-38  
 Bring to Front Tool, 1-17, 2-34  
 Bringing Objects to the Front, 2-35  
 Browse for Folder dialog box, 2-14, 2-16, 2-53  
 Button Objects, 2-2

## C

Cells, 2-3  
 Breaking, 1-18, 2-3, 2-38  
 Making, 1-18, 2-3, 2-38  
 Change Password dialog box, 2-90  
 ChangePassword(), 5-41  
 Changing  
 Security Log On Password, 2-90  
 Size of a floating Toolbar, 1-12  
 System time, 2-102  
 Check Box Control Wizard, 2-56, 2-59  
 Choose ActiveX Script dialog box, 2-78  
 Choose Key dialog box, 4-3, 5-13  
 Choose Name dialog box, 4-4  
 Choose tag type dialog box, 3-7  
 Clearing Links, 2-24  
 Close Window Tool, 1-13, 2-13  
 Close WindowMaker on Transfer to WindowViewer,  
 2-6  
 Close WindowViewer on Transfer to WindowMaker,  
 2-81

Closing windows, 2-13  
 Color Palettes  
 Blotter Tool, 1-24, 2-44  
 Classic Colors, 1-21  
 Creating custom colors, 1-22  
 Hue, 1-23  
 Importing/Export Custom palettes, 1-24  
 Luminosity, 1-23  
 Saturation, 1-23  
 Standard, 1-21  
 Color Tools  
 Fill, 1-15  
 Line, 1-15  
 Transparent Color, 1-15  
 Window Background, 1-15  
 Combo Box Control Wizard, 2-56, 2-58  
 Common Window Dialog Box Features, 1-29  
 Communication Protocols  
 DDE, 8-2  
 FastDDE, 8-2  
 NetDDE, 8-2  
 SuiteLink, 8-2  
 Comparisons ( <, >, <=, >=, ==, <> ) Operator, 5-31  
 Complement ( ~ ) Operator, 5-30  
 Complex Objects, 2-3  
 ActiveX Controls, 2-4  
 Buttons, 2-3  
 Cells, 2-3  
 Symbols, 2-3  
 Trends, 2-3  
 Wizards, 2-4  
 Complex Scripts, 5-32  
 Condition Scripts, 5-2  
 On False, 5-14, 5-15  
 On True, 5-14, 5-15  
 While False, 5-14, 5-15  
 While True, 5-14, 5-15  
 Condition Scripts Editor dialog box, 5-14  
 Configure Users command, 2-89  
 Configure Users dialog box, 2-89  
 Configuring  
 A Windows Control Wizard, 2-60  
 ActiveX Control, 2-68  
 ActiveX Control Properties, 2-69  
 Alarm Server Node, 2-82  
 Alarm/Events Printing, 6-17  
 Blink object speeds, 2-82  
 DDE Shares, 8-2  
 Distributed Alarm Object, 6-16  
 Dynamic Resolution Conversion, 2-94  
 Grid, 2-39  
 Operator's password, 2-90  
 Operator's security level, 2-89  
 Real-time Trend Object, 7-2  
 Standard Alarm Object, 6-11  
 System inactivity, 2-92  
 System Privileges, 2-99  
 Title bar text, 2-85  
 WindowMaker Properties, 2-5  
 WindowViewer, 2-80  
 WindowViewer as an NT Service, 2-96  
 WindowViewer Windows, 2-83

- Control Name, 2-56, 2-61
- Controlling Horizontal and Vertical Spacing, 2-35
- Convert to screen video resolution, 2-95
- Converting Placeholder Tagnames, 3-38
- Copy Object Tool, 1-13, 2-22
- Copying
  - Links, 2-24
  - Objects to the Windows Clipboard, 2-22
- Cos(), 5-39
- Creating
  - A new window, 2-8
  - Access Names, 8-7
  - ActiveX Event Scripts, 2-77, 5-18
  - Alarm Groups, 6-4
  - Animation Links, 4-1
  - Cells and Symbols, 2-37
  - Color Links, 4-17
  - Custom Color Palette, 1-22
  - Custom Security Log on Window, 2-91
  - Database Input File, 3-55
  - Database Record Templates, 3-58
  - Display Links
    - Color Links
      - Analog, 4-17
      - Analog Alarm, 4-17, 4-20
      - Analog Color Fill, 4-18
      - Discrete, 4-17
      - Discrete Alarm, 4-17, 4-19
      - Discrete Color Fill, 4-17
    - Fill Color, 4-17, 4-18, 4-19, 4-20
    - Line Color, 4-17, 4-18, 4-19, 4-20
    - Location, 4-17, 4-22
    - Miscellaneous, 4-17
      - Blink, 4-25
      - Disable, 4-25, 4-28
      - Orientation, 4-25, 4-26
      - Visibility, 4-25
    - Object Size, 4-17, 4-21
    - Percent Fill, 4-17, 4-23
    - Text Color, 4-17, 4-18, 4-19, 4-20
    - Value Display, 4-17
      - Analog, 4-29, 4-30
      - Discrete, 4-29
      - String, 4-29, 4-30
  - I/O Items, 8-10
  - Slider Touch Links
    - Discrete, 4-11
  - Standard Alarm Object, 6-10
  - Text Objects, 2-46
  - Touch Pushbutton Touch Links
    - Action, 4-12, 4-14
    - Discrete Value, 4-12, 4-13
    - Hide Window, 4-13, 4-16
    - Show Window, 4-13, 4-16
  - Transparent Bitmaps, 2-43
  - User Input Touch Links, 4-8
    - Analog, 4-8, 4-9
    - Discrete, 4-8, 4-9
    - String, 4-8, 4-10
- Cross Reference Utility, 3-24
  - Changing search criteria, 3-27
  - Creating Search Filters, 3-24

- Cross referencing by Tagname, 3-27, 3-29
- Filter wildcards, 3-25
- Icons, 3-25
- Printing Cross Reference files, 3-31
- Saving Cross Reference files, 3-30
- Search for all occurrences, 3-24
- Search for specific occurrences, 3-24
- Viewing search results, 3-25
- CSV File to Dump To dialog box, 3-52
- CSV File to Load From dialog box, 3-55
- Custom Color Palettes, 1-24
- Customizing
  - Resolution, 2-94
  - Your Development Environment, 2-5
  - Your Runtime Environment, 2-80
- Cut Object Tool, 1-13, 2-22
- Cutting
  - Object Links, 2-24
  - Objects to the Windows Clipboard, 2-22

## D

- Data Change Scripts, 5-2, 5-16
- Data Change Scripts Editor dialog box, 5-16
- Database Input File
  - Format, 3-55
  - Operating Modes, 3-56
- Daylight Savings, 2-102
- DBDump, 3-51
  - CSV Files, 3-52
  - Group output by types, 3-52
- DBLoad, 3-51, 3-53
  - Database Input File, 3-55
  - Database Input File Format, 3-55
  - Database Record Templates
    - Resetting Field Value Defaults, 3-58
    - Setting Field Value Defaults, 3-58
    - Type and Keyword Entries, 3-58
  - Keywords, 3-58
  - Operating Modes
    - MODE=ASK, 3-56
    - MODE=IGNORE, 3-57
    - MODE=REPLACE, 3-56
    - MODE=TERMINATE, 3-57
    - MODE=TEST, 3-57
    - MODE=UPDATE, 3-56
- DDE
  - Poke (one-time). *see* WWPoke()
- DDE Shares, 8-2
- Deadband, 3-10
- Debug Scripts, 2-82
- Decreasing a Rounded Object's Radius, 2-25
- Define Tag Filter dialog box, 3-22
- Defining
  - Analog Tagname Alarm Conditions, 3-15
  - Cross Reference search criteria, 3-24
  - Discrete Tagname Alarm Conditions, 3-15
  - I/O Items, 8-10
  - Item Names, 8-12
  - New Tagname, 3-6
  - Tag Browser Search Filters, 3-21
  - Tagname Alarm Conditions, 3-15, 6-8



- Tagname Details, 3-9
  - I/O Analog, 3-12
  - I/O Discrete, 3-11
  - I/O Message, 3-14
  - Memory Analog, 3-9
  - Memory Discrete, 3-9
  - Memory Message, 3-10
- Deleting
  - Access Names, 8-8
  - Alarm Groups, 6-6
  - Graphic objects, 2-24
  - Tag Browser Filters, 3-23
  - Tagnames from the Dictionary, 3-34
  - Unused Tagnames, 3-35
  - Windows, 2-13
- Deselecting
  - A Group of Selected Objects, 2-20
- Development Fast Switch, 2-7
- Deviation Deadband %, 3-16
- Diagnostics of Data Throughput, 8-2
- Diagonal Line Tool, 1-15
- Dialog Boxes
  - About, 1-34
  - Access Names, 8-7
  - ActiveX Control Browser, 2-72, 5-20
  - ActiveX Control Installation, 2-65
  - ActiveX Control Properties, 2-68
  - ActiveX Event Scripts Editor, 5-19
  - Add a Color, 1-22
  - Add Access Names, 8-7
  - Alarm Configuration, 6-11
  - Alarm Properties, 6-16
  - Alarm Properties - Printing, 6-17
  - Animation Selection, 4-2
  - App Development, 2-97
  - Application Action Script, 5-9
  - Application Directory to Dump From, 3-54
  - Browse for Folder, 2-14, 2-16, 2-53
  - Change Password, 2-90
  - Choose ActiveX Script, 2-78
  - Choose Key, 4-3, 5-13
  - Choose Name, 4-4
  - Choose tag type, 3-7
  - Common features, 1-29
  - Condition Scripts Editor, 5-14
  - Configure Users, 2-89
  - CSV File to Dump To, 3-52
  - CSV File to Load From, 3-55
  - Data Change Scripts Editor, 5-16
  - Define Tag Filter, 3-22
  - Details view, 1-30
    - auto-sizing columns, 1-30
    - sizing columns, 1-30
    - sort order, 1-30
  - Discrete Alarm State, 3-15
  - Duplicate Name (DBLoad), 3-56
  - Edit Custom Color, 2-44
  - Font, 2-48
  - Format Alarm Message, 6-13
  - Home Windows, 2-85
  - Initializing I/O, 2-82
  - InTouch Cross Reference Search Criteria, 3-24
  - InTouch Cross Reference View Options, 3-27
  - Log On, 2-90
  - Node Properties
    - App Development, 2-97
    - Resolution, 2-95
  - Object Type, 4-2
  - Problem with Export Operation, 2-15
  - Real Time Trend Configuration, 7-2
  - Remove Wizard from Toolbar, 2-55, 2-67
  - Replace Text, 2-49
  - Resolution, 2-95
  - Save Window, 2-14
  - Script Editor
    - Application Scripts, 5-9
    - Key Scripts, 5-11
    - Touch Action Scripts, 5-12
    - Windows Scripts, 5-10
  - Services, 2-97
  - Substitute Strings, 2-48, 2-49
  - Substitute Tags, 3-37
  - Tagname Dictionary, 3-6
    - Alarms, 3-6
    - Details, 3-6
      - I/O Analog, 3-12
      - I/O Discrete, 3-11
      - I/O Message, 3-14
      - Memory Analog, 3-10
      - Memory Discrete, 3-9
      - Memory Message, 3-11
    - Main, 3-6, 3-7
    - Members, 3-6
  - Window Configuration, 2-83
  - Window Properties, 2-8
  - WindowMaker Printout, 1-14, 5-37
  - WindowMaker Properties, 2-5
  - Windows to Export, 2-14
  - Windows to Print, 3-33
  - Windows to Show when touched, 4-16
  - WindowViewer Properties, 2-80
  - Wizard Selection, 1-14, 2-53, 2-66
  - Wizard/ActiveX Installation, 2-52
  - Wonderware Service User, 2-99
- DialogStringEntry(), 5-42
- DialogValueEntry(), 5-42
- Direct Pushbutton, 4-13
- Disable
  - Links, 4-25, 4-28
- Disabling
  - ALT key in WindowViewer, 2-85
  - CTRL-ESC key, 2-85
- Discrete Alarm State dialog box, 3-15
- Discrete Input Links, 4-9
- Discrete Value Display Links, 4-29
- Discrete Value Touch Pushbutton Links, 4-12, 4-13
- Display Links, 4-17
  - Color Links
    - Analog, 4-17
    - Analog Alarm, 4-17, 4-20
    - Analog Color Fill, 4-18
    - Discrete, 4-17
    - Discrete Alarm, 4-17, 4-19
    - Discrete Color Fill, 4-17

- Fill Color, 4-17, 4-18, 4-19, 4-20
- Line Color, 4-17, 4-18, 4-19, 4-20
- Location, 4-17, 4-22
- Miscellaneous, 4-17
  - Blink, 4-25
  - Disable, 4-25, 4-28
  - Orientation, 4-25, 4-26
  - Visibility, 4-25
- Object Size, 4-17, 4-21
- Percent Fill, 4-17, 4-23
- Text Color, 4-17, 4-18, 4-19, 4-20
- Value Display, 4-17
  - Analog, 4-29, 4-30
  - Discrete, 4-29
  - String, 4-29, 4-30
- Displaying
  - Numeric values, 2-45, 2-46
  - Tag Usage Count, 3-36
- Distributed
  - Applications and Time Zones, 2-101
- Distributed Alarm Object
  - Configuring, 6-16
- Division ( / ) Operator, 5-30
- Docked Toolbars, 1-12
- Draw Object Toolbar, 1-15, 2-18
  - 3-D Button Tool, 1-16
  - Bitmap Tool, 1-16, 2-40, 2-42, 2-43
  - Diagonal Line Tool, 1-15
  - Ellipse Tool, 1-15
  - Horizontal/Vertical Line Tool, 1-15
  - Polygon Tool, 1-16
  - Polyline Tool, 1-16
  - Real time Trend Tool, 1-16, 7-2
  - Rectangle Tool, 1-15
  - Rounded Rectangle Tool, 1-15
  - Selector Tool, 1-15
  - Text Object Tool, 1-16, 2-46
- Drop Down Combo Box, 2-59
- Drop Down List Combo Box, 2-59
- DText(), 5-38
- Duplicate Name dialog box, 3-56
- Duplicate Object Tool, 1-13, 2-21
- Duplicating
  - Graphic objects, 2-21
  - Windows, 2-13

## E

- Edit Custom Color dialog box, 2-44
- Editing
  - Text Objects, 2-48
- Ellipse Tool, 1-15
- Ellipsis, 1-32
- Enable Fast Switch, 2-7
- Engineering Units, 3-10
- Enlarge Font Tool, 1-15
- Erasing
  - Graphic objects, 2-24
- Error Messages
  - Scripts, 5-46
  - Windows Controls Script Functions, 5-50
- ErrorNumber, 5-50

- Evenly spacing objects, 2-35
- Events
  - Formatting Alarm/Event Message, 6-13
- Exclusive OR ( ^ ) Operator, 5-31
- Exiting a Control Structure, 5-26
- Exp(), 5-39
- Exporting
  - Custom Color Palette, 1-25
- Exporting Windows, 2-14
  - Placeholder Tagnames, 2-15, 2-17
  - Problems, 2-15
- Extended Tagname Support, 3-2, 3-5

## F

- Fast Switch, 2-7, 2-81
- FileCopy(), 5-40
- FileDelete(), 5-40
- FileMove(), 5-40
- FileReadFields(), 5-40
- FileReadMessage(), 5-41
- FileWriteFields(), 5-41
- FileWriteMessage(), 5-41
- Fill Color Tool, 1-15
- Filled Objects, 2-2
- Five Second Time Limit, 5-24
- Flip Horizontal Tool, 1-18, 2-36
- Flip Vertical Tool, 1-18, 2-36
- Flipping Cells, 2-38
- Flipping Objects, 2-36
  - Horizontally, 1-18, 2-36
  - Vertically, 1-18, 2-36
- Floating a docked Toolbar, 1-12
- Floating Toolbars, 1-12
- Font dialog box, 2-48
- Font Point Size, 2-48
- Font Tool, 1-14, 2-47
- Format Alarm Message dialog box, 6-13
- Format Toolbar, 1-14, 2-45
  - Bold Tool, 1-14
  - Centered Tool, 1-15
  - Enlarge Font Tool, 1-15
  - Fill Color Tool, 1-15
  - Font Tool, 1-14, 2-47
  - Italic Tool, 1-14
  - Left Justified Tool, 1-15
  - Line Color Tool, 1-15
  - Reduce Font Tool, 1-15
  - Right Justified Tool, 1-15
  - Text Color Tool, 1-15
  - Transparent Tool, 1-15
  - Underline Tool, 1-15
  - Window Background Color Tool, 1-15
- Formatting Text Objects, 2-45
- FOR-NEXT Loop Scripts, 5-23, 5-32
  - Exiting a Control Structure, 5-26
  - Five Second Time Limit, 5-24
  - Loop Variable Value After Loop Execution, 5-25
  - Nested Control Structures, 5-25
  - Nesting FOR-NEXT Loops, 5-24
  - Screen Updates, 5-24
- Frame Styles, 2-10

Full Screen Mode Tool, 1-16

## Functions

### Distributed Alarm System

#### Alarm Acknowledge Functions

Ack, 5-41

Inserting into a script, 5-4

### Math Functions, 5-39

Abs(), 5-39

ArcCos(), 5-39

ArcSin(), 5-39

ArcTan(), 5-39

Cos(), 5-39

Exp(), 5-39

Int(), 5-39

Log(), 5-39

LogN(), 5-39

Pi(), 5-39

Round(), 5-39

Sgn(), 5-40

Sin(), 5-40

Sqrt(), 5-40

Tan(), 5-40

Trunc(), 5-40

### Misc Functions, 5-41

Ack(), 5-41

ChangePassword(), 5-41

DialogStringEntry(), 5-42

DialogValueEntry(), 5-42

GetNodeName(), 5-42

GetPropertyD(), 5-42

GetPropertyI(), 5-43

GetPropertyM(), 5-43

Hide(), 5-43

HideSelf(), 5-43

IOSetAccessName, 5-43

IOSetItem(), 5-43

LogMessage(), 5-43

PlaySound(), 5-43

PrintWindow(), 5-43

RestartWindowViewer(), 5-43

SendKeys(), 5-43

SetDdeAppTopic(), 5-43

SetDdelItem(), 5-43

SetPropertyD(), 5-43

SetPropertyI(), 5-43

SetPropertyM(), 5-43

Show(), 5-43

ShowAt(), 5-43

ShowHome(), 5-44

ShowTopLeftAt(), 5-44

wcAddItem(), 5-44

wcClear(), 5-44

wcDeleteItem(), 5-44

wcDeleteSelection(), 5-44

wcErrorMessage(), 5-44

wcFindItem(), 5-44

wcGetItem(), 5-44

wcGetItemData(), 5-44

wcInsertItem(), 5-44

wcLoadList(), 5-44

wcLoadText(), 5-44

wcSaveList(), 5-44

wcSaveText(), 5-44

wcSetItemData(), 5-44

WWControl(), 5-45

WWExecute(), 5-45

WWPoke(), 5-45

WWRequest(), 5-45

### String Functions, 5-38

DText(), 5-38

StringASCII(), 5-38

StringChar(), 5-38

StringFromIntg(), 5-38

StringFromReal(), 5-38

StringFromTime(), 5-38

StringInString(), 5-38

StringLeft(), 5-38

StringLen(), 5-38

StringLower(), 5-38

StringMid(), 5-38

StringReplace(), 5-38

StringRight(), 5-38

StringSpace(), 5-38

StringTest(), 5-39

StringToIntg(), 5-39

StringToReal(), 5-39

StringTrim(), 5-39

StringUpper(), 5-39

Text(), 5-39

### System Functions, 5-40

ActivateApp(), 5-40

FileCopy(), 5-40

FileDelete(), 5-40

FileMove(), 5-40

FileReadFields(), 5-40

FileReadMessage(), 5-41

FileWriteFields(), 5-41

FileWriteMessage(), 5-41

InfoAppActive(), 5-41

InfoAppTitle(), 5-41

InfoDisk(), 5-41

InfoFile(), 5-41

InfoInTouchAppDir(), 5-41

InfoResources(), 5-41

StartApp(), 5-41

## G

General Toolbar, 1-13, 2-8, 2-18

Close Window Tool, 1-13, 2-13

Copy Object Tool, 1-13, 2-22

Cut Object Tool, 1-13, 2-22

Duplicate Object Tool, 1-13, 2-21

New Window Tool, 1-13

Open Window Tool, 1-13, 2-12

Paste Object Tool, 1-14, 2-23

Print Tool, 1-14

Redo Tool, 1-14, 2-20

Save All Windows Tool, 1-13

Save Window Tool, 1-13, 2-12

Undo Tool, 1-14, 2-19, 2-20

GetNodeName(), 5-42

GetPropertyD(), 5-42

GetPropertyI(), 5-43

GetPropertyM(), 5-43  
 GMT, 2-101  
 Graphic Object Tools  
   3-D Button, 1-16  
   Bitmap, 1-16, 2-40, 2-42, 2-43  
   Diagonal Line, 1-15  
   Ellipse, 1-15  
   Horizontal/Vertical Line, 1-15  
   Polygon, 1-16  
   Polyline, 1-16  
   Real time Trend, 1-16, 7-2  
   Rectangle, 1-15  
   Rounded Rectangle, 1-15  
   Selector, 1-15  
   Text Object, 1-16, 2-46  
 Graphic Objects  
   Adding a Polygon Point, 2-26  
   Aligning, 2-29  
   Animating, 4-6  
   Applying color, 1-21  
   Blink speed, 2-82  
   Cells, 2-37  
   Cells - Flipping, 2-38  
   Clearing Links, 2-24  
   Combining, 1-18, 2-3, 2-38  
   Copying, 2-22  
   Copying Links, 2-24  
   Cutting, 2-22  
   Decreasing Radius - Short Cut, 2-25  
   Deleting a Polygon Point, 2-26  
   Deleting a Polyline Point, 2-26  
   Deleting/Erasing, 2-24  
   Deselecting a Group, 2-20  
   Duplicating, 2-21  
   Enlarging Radius, 2-25  
   Enlarging Radius - Short Cut, 2-25  
   Flipping, 2-36  
   Layering, 2-34  
   Pasting, 2-23  
   Pasting Links, 2-24  
   Reducing Radius, 2-25  
   Removing outlines, 2-51  
   Reshaping a Polygon, 2-26  
   Reshaping a Polyline, 2-26  
   Rotating, 2-35  
   Selecting, 2-19  
   Selecting a Group, 2-20  
   Selecting All, 2-20  
   Selecting Multiple, 2-20  
   Sizing, 2-19  
   Snapping to Grid, 2-38  
   Spacing Horizontally, 2-35  
   Spacing Vertically, 2-35  
   Symbols, 2-37  
 Group Var Tag Type, 3-4

## H

Help for Script Functions, 5-5  
 Hide All Toolbars Tool, 1-16  
 Hide Window Touch Pushbutton Links, 4-13, 4-16  
 Hide(), 5-43

HideSelf(), 5-43  
 Hiding all Toolbars, 1-13  
 Hiding Cursor in WindowViewer, 2-85  
 Hiding the Title and Menu Bars, 2-11  
 Historical .Fields  
   .UpdateCount, 3-49  
 Historical Trending, 7-1  
 Home Windows, 2-85  
 Home Windows dialog box, 2-85  
 Horizontal Percent Fill Links, 4-23  
 Horizontal Slider Links, 4-11  
 Horizontal/Vertical Line Tool, 1-15  
 Hue, 1-23

## I

I/O  
   Discrete Tag Type, 3-3  
   Integer Tag Type, 3-3  
   Message Tag Type, 3-4  
   Real Tag Type, 3-4  
   Tag Types, 3-3  
 I/O Addressing, 8-6  
 I/O Communications, 8-1  
 I/O Communications Error, 2-82  
 I/O Servers  
   Monitoring Status of Server Communications, 8-17  
 I/OStatus Topic Name, 8-13  
 IF-THEN-ELSE Statement, 5-32  
 Images  
   Importing, 2-40  
   JPEG, 2-40  
   PCX, 2-40  
   TGA, 2-40  
 Importing  
   ActiveX Event Scripts, 2-79  
   Bitmaps, 2-40  
   Custom Color Palette, 1-24  
   Images, 2-40  
   InTouch QuickScripts, 5-35  
   Placeholder Tagnames, 5-36  
   Window Scripts, 2-17  
   Windows, 2-16, 2-17  
 Inclusive OR (|) Operator, 5-31  
 Increasing a Rounded Object's Radius, 2-25  
 Increasing Real-time Trending Performance, 7-4  
 InfoAppActive(), 5-41  
 InfoAppTitle(), 5-41  
 InfoDisk(), 5-41  
 InfoFile(), 5-41  
 InfoInTouchAppDir(), 5-41  
 InfoResources(), 5-41  
 Initializing I/O dialog box, 2-82  
 Installing  
   ActiveX Controls, 2-65  
   Wizards, 2-52  
 Instrument Failure Monitoring, 3-39  
 Int(), 5-39  
 Internal System \$Tagnames, 3-40  
   \$AccessLevel, 3-40  
   \$AlarmPrinterError, 3-40  
   \$AlarmPrinterNoPaper, 3-40

- \$AlarmPrinterOffline, 3-40
- \$AlarmPrinterOverflow, 3-40
- \$ApplicationChanged, 3-40
- \$ApplicationVersion, 3-40
- \$ChangePassword, 3-40
- \$ConfigureUsers, 3-40
- \$Date, 3-41
- \$DateString, 3-41
- \$DateTime, 3-41
- \$Day, 3-41
- \$Hour, 3-41
- \$InactivityWarning, 3-41
- \$Minute, 3-41
- \$Month, 3-41
- \$Msec, 3-41
- \$NewAlarm, 3-41
- \$ObjHor, 3-41
- \$ObjVer, 3-41
- \$OperatorEntered, 3-42
- \$PasswordEntered, 3-42
- \$Second, 3-42
- \$StartDdeConversations, 3-42
- \$System, 3-42
- \$Time, 3-42
- \$TimeString, 3-42
- \$Year, 3-42
- InTouch Access Names
  - Creating, 8-7
  - Deleting, 8-8
  - Modifying, 8-8
- InTouch Cross Reference Search Criteria dialog box, 3-24
- InTouch Cross Reference View Options dialog box, 3-27
- InTouch DDE Address, 8-6
- IOSetAccessName, 5-43
- IOSetItem, 5-43
- Italic Tool, 1-14
- Item Name, 8-5

## K

- Key
  - Application, 5-2
- Key Equivalents, 4-3, 5-13
- Key Scripts, 5-2
  - On Key Down, 5-11
  - On Key Up, 5-11
  - While Down, 5-11
- Key Scripts Editor dialog box, 5-11
- Keyboard Arrow Keys, 1-33
- Keyboard Short Cuts, 1-32

## L

- Layering Objects, 2-34
- Left Justified Tool, 1-15
- Levels of Undo, 1-14, 2-7
- License Viewing Utility, 1-34
- Line Color Tool, 1-15
- Line Menu, 2-51
- Line Objects, 2-2

- Line Selection Precision, 2-7
- List Box Control Wizard, 2-56, 2-58
- Local Tag Count, 2-6, 3-5, 3-34, 3-36
- Local Variables, 5-21
  - Data-types, 5-22
  - Valid Syntax, 5-21
- Location Links, 4-22
- Log Events, 3-8
- Log On dialog box, 2-90
- Log On Window, 2-91
- Log(), 5-39
- Logging
  - Off an Application, 2-91
  - On to an application, 2-90
- LogMessage(), 5-43
- LogN(), 5-39
- Loop Scripts
  - FOR-NEXT, 5-32
- Loop Variable Value After Loop Execution, 5-25
- Luminosity, 1-23

## M

- Major Deviation, 3-16
- Make Cell Tool, 1-18, 2-38
- Make Symbol Tool, 1-18, 2-38
- Math Functions, 5-39
- Math Script Functions, 5-5
- Memory
  - Discrete Tag Type, 3-3
  - Integer Tag Type, 3-3
  - Message Tag Type, 3-3
  - Real Tag Type, 3-3
- Microsoft *Network DDE*, 8-2
- Minimum Memory to keep free, 2-81
- Minor Deviation, 3-16
- Mirroring Objects, 2-36
- Misc Functions, 5-41
  - IOAccessName, 5-43
- Miscellaneous Links
  - Blink, 4-25
  - Disable, 4-25, 4-28
  - Orientation, 4-25, 4-26
  - Visibility, 4-25
- Miscellaneous Script Functions, 5-5
- Miscellaneous Tag Types, 3-4
  - Group Var, 3-4
- Modifying
  - Access Names, 8-8
  - Alarm Groups, 6-6
- Modulo (MOD) Operator, 5-30
- Monitoring
  - Multiple Input Device Status, 8-19
  - Status of a I/O Conversation, 8-13
  - Status of Server Communications, 8-17
- Mouse short cuts, 1-31
- Moving Objects with the Arrow Keys, 1-33
- Multiplication ( \* ) Operator, 5-30

## N

- Navigating in the Application Explorer, 1-5

Negation ( - ) Operator, 5-30  
 Nested Control Structures, 5-25  
 Nesting FOR-NEXT Loops, 5-24  
 NetDDE, 8-2  
 Network Application Development  
   Configuring DDE Shares, 8-2  
   DDE Access Names, 8-8  
 New Window Tool, 1-13  
 Node Properties dialog box, 2-95, 2-97  
 NOT Operator, 5-31  
 NT Network DDE, 8-3  
 NT Services, 8-4  
 NT User Manager, 2-99

## O

Object Size Links, 4-21  
 Object Type dialog box, 4-2  
 Objects  
   ActiveX Controls, 2-4  
   Adding a Polygon Point, 2-26  
   Aligning, 2-29  
   Animating, 4-6  
   Attributes, 2-2  
   Bitmaps, 2-3  
   Blink speed, 2-82  
   Buttons, 2-2  
   Cells, 2-3, 2-37  
   Cells - Flipping, 2-38  
   Clearing Links, 2-24  
   Combining, 2-37  
   Complex, 2-3  
   Copying, 2-22  
   Copying Links, 2-24  
   Cutting, 2-22  
   Decreasing Radius - Short Cut, 2-25  
   Deleting a Polygon Point, 2-26  
   Deleting a Polyline Point, 2-26  
   Deleting/Erasing, 2-24  
   Deselecting a Group, 2-20  
   Duplicating, 2-21  
   Enlarging Radius, 2-25  
   Enlarging Radius - Short Cut, 2-25  
   Filled Shapes, 2-2  
   Flipping, 2-36  
   Layering, 2-34  
   Line, 2-2  
   Pasting, 2-23  
   Pasting Links, 2-24  
   Reducing Radius, 2-25  
   Removing outlines, 2-51  
   Reshaping a Polygon, 2-26  
   Reshaping a Polyline, 2-26  
   Rotating, 2-35  
   Selecting, 2-19  
   Selecting a Group, 2-20  
   Selecting All, 2-20  
   Selecting Multiple, 2-20  
   Simple, 2-2  
   Sizing, 2-19  
   Snapping to Grid, 2-38  
   Spacing Horizontally, 2-35

Spacing Vertically, 2-35  
 Symbols, 2-3, 2-37  
 Text, 2-2  
 Trends, 2-3  
 Wizards, 2-4  
 OCXs, 2-64  
 OLE Controls, 2-64  
 On False Condition Scripts, 5-14, 5-15  
 On Hide Window Scripts, 5-10  
 On Key Down Scripts, 5-11, 5-12  
 On Key Up Scripts, 5-11, 5-12  
 On Show Window Scripts, 5-10  
 On Shutdown Application Scripts, 5-9  
 On StartUp Application Scripts, 5-9  
 On True Condition Scripts, 5-14, 5-15  
 On While Down Key Scripts, 5-11, 5-12  
 Open Window Tool, 1-13, 2-12  
 Opening windows, 2-12  
 Operands, 5-28, 5-29  
 Operations  
   Addition and Concatenation, 5-29  
   Assignment, 5-29  
   Bitwise AND, 5-29  
   Complement, 5-29  
   Division, 5-29  
   Equivalency, 5-29  
   Exclusive OR, 5-29  
   Greater than, 5-29  
   Greater than or Equal to, 5-29  
   Inclusive OR, 5-29  
   Left Shift, 5-29  
   Less than, 5-29  
   Less than or Equal to, 5-29  
   Logical NOT, 5-29  
   Logical OR, 5-29  
   Modulo, 5-29  
   Multiplication, 5-29  
   Negation, 5-29  
   Not Equal to, 5-29  
   Power, 5-29  
   Right Shift, 5-29  
   Subtraction, 5-29  
 Operator Descriptions  
   Addition ( + ), 5-30  
   AND, OR, and NOT, 5-31  
   Assignment ( = ), 5-31  
   Bitwise AND ( & ), 5-31  
   Comparisons ( <, >, <=, >=, ==, <> ), 5-31  
   Complement ( ~ ), 5-30  
   Division ( / ), 5-30  
   Exclusive OR ( ^ ), 5-31  
   Inclusive OR ( | ), 5-31  
   Modulo (MOD), 5-30  
   Multiplication ( \* ), 5-30  
   Negation ( - ), 5-30  
   Parentheses ( ), 5-30  
   Power ( \*\* ), 5-30  
   Shift Left (SHL), 5-31  
   Shift Right (SHR), 5-31  
   Subtraction ( - ), 5-30  
 Operator Precedence  
   Highest Precedence, 5-29

Lowest Precedence, 5-29  
 Optimize performance for memory, 2-81  
 Option Buttons, 2-56, 2-60  
 OR Operator, 5-31  
 Orientation Links, 4-25, 4-26  
 Overlay type window, 2-9

## P

**Palettes**  
 Blotter Tool, 1-24, 2-44  
 Creating custom colors, 1-22  
 Custom Color, 1-24  
 Importing/Exporting, 1-24  
 Parentheses ( ) Operator, 5-30  
 Paste Object Tool, 1-14, 2-23  
**Pasting**  
 ActiveX Control in a window, 2-66  
 Bitmap from the Windows Clipboard, 2-42  
 Graphic Objects, 2-23  
 Object Links, 2-24  
 Objects from the Windows Clipboard, 2-23  
 Wizard in a window, 2-53  
 Percent Fill Links, 4-23  
 Pi(), 5-39  
 Pick Through Hollow Objects, 2-6  
 Placeholder tagnames, 2-15, 2-17  
 Placeholder Tagnames, 3-38, 5-36  
 PlaySound(), 5-43  
 Polygon Tool, 1-16  
**Polygons**  
 Adding a point, 2-26  
 Deleting a point, 2-26  
 Reshaping, 1-18, 2-26  
 Polyline Tool, 1-16  
**Polylines**  
 Adding a point, 2-26  
 Deleting a point, 2-26  
 Reshaping, 1-18, 2-26  
 Popup type window, 2-9  
 Power ( \*\* ) Operator, 5-30  
 Precision alignment, 1-19  
 Print Tool, 1-14  
**Printing**  
 Alarms/Events, 6-17  
 Cross Reference Files, 3-31  
 Database information, 1-14  
 Scripts, 5-37  
 Tagname Dictionary Details, 3-32  
 Window information, 1-14  
 PrintWindow(), 5-43  
 Problem with Export Operation, 2-15  
 Problem with Export Operation dialog box, 2-15  
**Program Elements**  
 Application Explorer, 1-4  
 Arrange Toolbar, 1-17  
 Color Palette, 1-21, 2-44  
 Draw Object Toolbar, 1-15  
 Floating/Docking Toolbars, 1-11  
 Format Toolbar, 1-14  
 General Toolbar, 1-13  
 Ruler, 1-19

Status Bar, 1-20  
 Toolbars, 1-11  
 View Toolbar, 1-16  
 Wizards/ActiveX Toolbar, 1-14  
 Promotional InTouch license, 2-12  
**Pushbuttons**  
 Direct, 4-13  
 Reset, 4-13  
 Reverse, 4-13  
 Set, 4-13  
 Toggle, 4-13

## Q

**QuickScripts**  
 Action Touch Pushbutton, 5-2  
 ActiveX Events, 5-2  
 Application, 5-2  
 Common Procedures, 5-3  
 Condition, 5-2  
 Data Change, 5-2  
 Importing, 2-17  
 Importing ActiveX Event Scripts, 2-79  
 Key, 5-2  
 Stopping in runtime, 2-84, 2-85  
 Window, 5-2  
 Windows, 2-11

## R

Radio Buttons, 2-56, 2-60  
 Real Time Trend Configuration Dialog Box, 7-2  
 Real-time Trending, 7-1  
 Configuring, 7-2  
 Increasing Performance, 7-4  
 Real-time Trend Object, 7-2  
 Rectangle Tool, 1-15  
**Redo**  
 Levels, 1-14, 2-7  
 Stacks, 2-7  
 Redo Tool, 1-14, 2-20  
 Redrawing an Object, 2-21  
 Reduce Font Tool, 1-15  
**Remote**  
 Alarm Nodes, 2-82  
 Tag Count, 2-6, 3-5, 3-34, 3-36  
 Remove Wizard from Toolbar dialog box, 2-55, 2-67  
**Removing**  
 ActiveX Control from the toolbar, 2-67  
 ActiveX Controls from an application, 2-65  
 Object Outlines, 2-51  
 Wizard from the toolbar, 2-54  
 Replace Text dialog box, 2-49  
 Replace Type Window, 2-9  
**Replacing**  
 Text, 2-49  
 Reset Pushbutton, 4-13  
 Reshape Object Tool, 1-18, 2-26  
 Reshaping a Polyline or Polygon Object, 2-26  
 Resolution dialog box, 2-95  
 RestartWindowViewer(), 5-43  
**Restoring**

- Scripts, 5-7
- Retentive Parameters, 3-8
- Retentive Value, 3-8
- Reusing ActiveX Event Scripts, 2-77
- Reverse Pushbutton, 4-13
- Right Justified Tool, 1-15
- Right-Click Menus, 1-26
  - Dialog Boxes, 1-27
  - Floating Toolbars, 1-27
  - Objects, 1-26, 2-18
  - Windows, 1-26, 2-8
- Rotate Clockwise Tool, 1-18, 2-36
- Rotate CounterClockwise Tool, 1-18, 2-36
- Rotating
  - Bitmaps, 2-40
  - Graphic Objects, 2-35
    - Clockwise, 1-18, 2-36
    - Counter-Clockwise, 1-18, 2-36
    - Flipping Horizontally, 1-18, 2-36
    - Flipping Vertically, 1-18, 2-36
  - Images, 2-40
  - Text Objects, 2-45
- Round(), 5-39
- Rounded Rectangle Tool, 1-15
- Ruler, 1-19
  - Pixel spacing, 1-19
  - Precision alignment, 1-19
  - Showing/Hiding, 1-19
  - Tick Marks, 1-19
- Ruler Tool, 1-17
- Runtime Environment
  - Customizing, 2-80
- Runtime Fast Switch, 2-7, 2-81

## S

- Saturation, 1-23
- Save All Windows Tool, 1-13
- Save Window dialog box, 2-14
- Save Window Tool, 1-13, 2-12
- Saving
  - Cross Reference Files, 3-30
  - Scripts, 5-7
  - windows, 2-12
- Scaling I/O Tagnames, 3-39
- Screen Updates, 5-24
- Script Editor
  - Common Procedures, 5-3
- Script Functions, 2-62, 2-63, 5-44
  - Help for, 5-5
  - Types
    - Add-ons, 5-5
    - All, 5-5
    - Math, 5-5
    - Misc, 5-5
    - String, 5-5
    - System, 5-5
- Scripts
  - Action Touch Pushbutton, 5-2
    - On Key Down, 4-14
    - On Key Up, 4-14
    - While Down, 4-14

- ActiveX Events, 5-2
- ActiveX Events Scripts, 5-18
- Application, 5-2
  - On Shutdown, 5-9
  - On Startup, 5-9
  - While Running, 5-9
- Assigning a Key Equivalent, 5-13
- Comparisons, 5-32
  - Equivalency, 5-32
  - Greater than, 5-32
  - Greater than or Equal to, 5-32
  - Less than, 5-32
  - Less than or Equal to, 5-32
  - Not Equal to, 5-32
- Complex Scripts, 5-32
- Condition Scripts, 5-2
  - On False, 5-14, 5-15
  - On True, 5-14, 5-15
  - While False, 5-14, 5-15
  - While True, 5-14, 5-15
- Creating New, 5-3
- Data Change Scripts, 5-2, 5-16
- Deleting, 5-4
- Exiting a Control Structure, 5-26
- FOR-NEXT, 5-32
- FOR-NEXT Loop Scripts, 5-23, 5-24, 5-25, 5-26
- Functions. *see* Functions
- IF-THEN-ELSE, 5-32
- Importing, 5-35
- Importing ActiveX Event Scripts, 2-79
- Indent/Unindenting text, 5-3
- Inserting
  - Functions, 5-4
  - Tagnames, 5-5
  - Window Names, 5-7
- Key Scripts
  - On Key Down, 5-11
  - On Key Up, 5-11
  - While Down, 5-11
- Keyword
  - AS, 5-21
  - DIM, 5-21
- Local Variables, 5-21
  - Data-types, 5-22
  - Valid Syntax, 5-21
- Loop Variable Value After Loop Execution, 5-25
- Nested Control Structures, 5-25
- Nesting FOR-NEXT Loops, 5-24
- Operands, 5-28, 5-29
- Operations
  - Addition and Concatenation, 5-29
  - Assignment, 5-29
  - Bitwise AND, 5-29
  - Complement, 5-29
  - Division, 5-29
  - Equivalency, 5-29
  - Exclusive OR, 5-29
  - Greater than, 5-29
  - Greater than or Equal to, 5-29
  - Inclusive OR, 5-29
  - Left Shift, 5-29
  - Less than, 5-29



- Less than or Equal to, 5-29
- Logical NOT, 5-29
- Logical OR, 5-29
- Modulo, 5-29
- Multiplication, 5-29
- Negation, 5-29
- Not Equal to, 5-29
- Power, 5-29
- Right Shift, 5-29
- Subtraction, 5-29
- Operator Descriptions
  - Addition ( + ), 5-30
  - AND, OR, and NOT, 5-31
  - Assignment ( = ), 5-31
  - Bitwise AND ( & ), 5-31
  - Comparisons ( <, >, <=, >=, ==, <> ), 5-31
  - Complement ( ~ ), 5-30
  - Division ( / ), 5-30
  - Exclusive OR ( ^ ), 5-31
  - Inclusive OR ( | ), 5-31
  - Modulo (MOD), 5-30
  - Multiplication ( \* ), 5-30
  - Negation ( - ), 5-30
  - Parentheses ( ), 5-30
  - Power ( \*\* ), 5-30
  - Shift Left (SHL), 5-31
  - Shift Right (SHR), 5-31
  - Subtraction ( - ), 5-30
- Operator Precedence
  - Highest Precedence, 5-29
  - Lowest Precedence, 5-29
- Printing Scripts, 5-37
- Required Syntax, 5-28
- Restoring, 5-7
- Saving, 5-7
- Screen Updates, 5-24
- Script Editor Error Messages, 5-46
- Simple Scripts, 5-28
- Stopping in runtime, 2-84, 2-85
- Styles and Syntax, 5-28
- Text
  - Clearing, 5-4
  - Copying, 5-4
  - Cutting, 5-4
  - Pasting, 5-4
  - Selecting All, 5-4
- Touch Pushbutton Action Scripts, 5-12
- Undoing last action, 5-4
- Validating, 5-7
- Window, 5-2
- Window Scripts
  - On Hide, 5-10
  - On Show, 5-10
  - While Showing, 5-10
- Windows, 2-11
- Security, 2-87
  - \$AccessLevel, 2-87, 2-91
  - \$ChangePassword, 2-90
  - \$ConfigureUsers, 2-91
  - \$Operator, 2-87
  - \$OperatorEntered, 2-91
  - \$PasswordEntered, 2-91
  - Access Levels, 2-87
  - Administrator, 2-87
  - Automatic Log-Off, 2-91
  - Changing a password, 2-90
  - Configuring inactivity, 2-92
  - Configuring Users, 2-89
  - Custom Log on Window, 2-91
  - Disable Animation Link, 2-89
  - Internal tagnames, 2-87
  - Logging Off, 2-91
  - Logging On, 2-90
  - Password, 2-87
- Selecting
  - All Objects in a Window, 2-20
  - Graphic Objects, 2-19
  - Group of Objects, 2-20
  - Home Windows, 2-85
  - Line Style, 2-51
  - Line Width, 2-51
  - Multiple Objects in a Window, 2-20
  - Text Object's Font, 2-48
  - Text Object's Font Point Size, 2-48
  - Windows to Import, 2-17
- Selector Tool, 1-15
- Send to Back Tool, 1-17, 2-34
- Sending Objects to the Back, 2-34
- SendKeys(), 5-43
- Services dialog box, 2-97
- Set Pushbutton, 4-13
- SetDdeAppTopic(), 5-43
- SetDdeItem(), 5-43
- SetPropertyD(), 5-43
- SetPropertyI(), 5-43
- SetPropertyM(), 5-43
- Setting WindowMaker's General Properties, 2-5
- Setting WindowViewer's General Properties, 2-80
- Setting your time zone, 2-101
- Sgn(), 5-40
- Shift Left (SHL) Operator, 5-31
- Shift Right (SHR) Operator, 5-31
- Short cuts and Accelerators, 1-32
- Show Tag Count, 2-6
- Show Tagname Count, 3-36
- Show Window Touch Pushbutton Links, 4-13, 4-16
- Show(), 5-43
- Show/Hide all Docked Toolbars, 1-16
- Show/Hide Docked Toolbars, 1-11
- Show/Hide Floating Toolbars, 1-12
- Show/Hide Ruler, 1-17
- Show/Hide Visible Grid, 1-17, 2-6, 2-39
- ShowAt(), 5-43
- ShowHome(), 5-44
- ShowTopLeftAt(), 5-44
- Simple Combo Box, 2-59
- Simple Objects, 2-2
  - Buttons, 2-2
  - Filled Shapes, 2-2
  - Line, 2-2
  - Text, 2-2
- Simple Scripts, 5-28
- Sin(), 5-40
- Sizing

- Graphic Objects, 2-19
- Slider links, 4-11
- Snap to Grid, 2-6, 2-38
  - Configuring the Grid, 2-39
- Snapping Objects, 2-38
  - Tool, 1-17, 2-39
- Space Horizontal Tool, 1-18, 2-35
- Space Vertical Tool, 1-18, 2-35
- Spacing Objects, 2-35
- Sqrt(), 5-40
- Standard Color Palette, 1-21
- Start Local Servers, 2-82
- StartApp(), 5-41
- Starting WindowViewer as an icon, 2-81
- Starting WindowViewer as an NT Service, 2-97
- Status Bar
  - Showing/Hiding, 1-20
- Stopping scripts in runtime, 2-84, 2-85
- String Functions, 5-38
- String Input Links, 4-10
- String Script Functions, 5-5
- String Value Display Links, 4-30
- StringASCII(), 5-38
- StringChar(), 5-38
- StringFromIntg(), 5-38
- StringFromReal(), 5-38
- StringFromTime(), 5-38
- StringInString(), 5-38
- StringLeft(), 5-38
- StringLen(), 5-38
- StringLower(), 5-38
- StringMid(), 5-38
- StringReplace(), 5-38
- StringRight(), 5-38
- StringSpace(), 5-38
- StringTest(), 5-39
- StringToIntg(), 5-39
- StringToReal(), 5-39
- StringTrim(), 5-39
- StringUpper(), 5-39
- Substitute Strings dialog box, 2-48, 2-49
- Substitute Tags dialog box, 3-37
- Substituting Tagnames, 3-37
- Subtraction ( - ) Operator, 5-30
- SuiteLink, 8-1
- Switching display modes, 1-16
- Symbol Objects, 2-3
- Symbols, 2-37
  - Breaking, 1-18, 2-38
  - Making, 1-18, 2-38
- System Script Functions, 5-5, 5-40

## T

- Tag Browser, 3-2, 3-17
  - Deleting Filters, 3-23
  - Filter wildcards, 3-22
  - Filtered Selection Mode, 3-17
  - Filters, 3-21
  - Selecting a .field, 3-43
  - Status Bar, 3-17
  - Unlimited Selection Mode, 3-17, 3-18

- Views, 3-19
  - Details, 3-20
  - List, 3-19
  - Tree, 3-21
- Tag License, 2-6, 3-5, 3-34, 3-36
- Tag Types
  - I/O, 3-3
    - I/O Discrete, 3-3
    - I/O Integer, 3-3
    - I/O Message, 3-4
    - I/O Real, 3-4
  - Memory, 3-3
    - Memory Discrete, 3-3, 3-9, 3-10, 3-11, 3-14
    - Memory Integer, 3-3, 3-9, 3-12
    - Memory Message, 3-3
    - Memory Real, 3-3
  - Miscellaneous, 3-4
    - Group Var, 3-4
- Tagname .Fields, 3-43
  - .Ack, 3-44, 6-20
  - .Alarm, 3-44, 6-20
  - .AlarmDevDeadband, 3-44, 6-20
  - .AlarmEnable, 6-20
  - .AlarmEnabled, 3-44
  - .AlarmValDeadband, 3-44, 6-20
  - .Comment, 3-45
  - .DevTarget, 3-45, 6-20
  - .HiHiLimit, 3-45, 6-20
  - .HiHiStatus, 3-45, 6-20
  - .HiLimit, 3-45, 6-20
  - .HiStatus, 3-45, 6-20
  - .LoLimit, 3-45, 6-20
  - .LoLoLimit, 3-45, 6-20
  - .LoLoStatus, 3-45, 6-20
  - .LoStatus, 3-45, 6-20
  - .MajorDevPct, 3-45, 6-20
  - .MajorDevStatus, 3-45, 6-20
  - .MaxRaw, 3-39
  - .MaxEU, 3-45
  - .MaxRange, 3-46
  - .MaxRaw, 3-46
  - .MinRaw, 3-39
  - .MinEU, 3-45
  - .MinorDevPct, 3-46, 6-20
  - .MinorDevStatus, 3-46, 6-20
  - .MinRange, 3-46
  - .MinRaw, 3-46
  - .Name, 3-46, 6-21
  - .Normal, 3-46, 6-21
  - .OffMsg, 3-45, 3-46
  - .OnMsg, 3-45, 3-46
  - .Quality, 3-47
  - .QualityLimit, 3-47
  - .QualityLimitString, 3-47
  - .QualityStatus, 3-47
  - .QualityStatusString, 3-47
  - .QualitySubstatus, 3-47
  - .QualitySubstatusString, 3-47
  - .RawValue, 3-47
  - .Reference, 3-47
  - .ReferenceComplete, 3-47
  - .ROCPct, 3-47, 6-21

- .ROCStatus, 3-47, 6-21
- .TimeDate, 3-48
- .TimeDateString, 3-48
- .TimeDateTime, 3-48
- .TimeDay, 3-48
- .TimeHour, 3-48
- .TimeMinute, 3-49
- .TimeMonth, 3-49
- .TimeMsec, 3-49
- .TimeSecond, 3-49
- .TimeTime, 3-49
- .TimeTimeString, 3-49
- .TimeYear, 3-49
- .Unack, 3-44, 3-49
- .UpdateCount, 3-49
- .Value, 3-49
- Inserting into a script, 5-6
- Selecting, 3-43
- Tagname Dictionary, 3-1
  - Extended Tagname Support, 3-2
  - Special Features, 3-2
  - Tag Browser, 3-2
  - Tag Types, 3-3
  - Tagname Cross Referencing, 3-2
- Tagname Dictionary dialog box, 3-6
- Tagname Dictionary Utilities, 3-51
  - DBDump, 3-51
  - DBLoad, 3-51
- Tagnames
  - .fields, 3-19
  - Alarm Groups, 3-7
  - Analog Alarm Details
    - % Deviation, 3-16
    - Alarm Types, 3-15
    - Deviation Deadband %, 3-16
    - Minor/Major Deviation, 3-16
    - Priority, 3-16
    - Rate of Change, 3-16
    - Target, 3-16
    - Value, 3-15
    - Value Deadband, 3-16
  - Analog Details
    - .Max EU, 3-10
    - .Min EU, 3-10
    - Deadband, 3-10
    - Eng Units, 3-10
    - Initial Value, 3-10
  - Auto-indexing, 3-6
  - Bit Fields, 3-49
  - Comments, 3-8
  - Conversion - Linear, 3-13
  - Conversion - Square Root, 3-13
  - Converting placeholders, 3-38
  - Cross Reference search criteria, 3-24
  - Cross Referencing, 3-2, 3-24
  - Defining
    - A new Tagname, 3-6
    - Alarm Conditions, 3-15
    - Details, 3-9
    - I/O types, 8-10
  - Deleting Tagnames, 3-34, 3-35
  - Discrete Alarm Details, 3-15
    - Alarm State, 3-15
    - Priority, 3-15
  - Displaying Use Counts, 3-36
  - Extended Support, 3-5
  - Finding in a script, 5-6
  - I/O Analog Details, 3-12
    - Access Name, 3-13
    - Conversion, 3-13
    - Deadband, 3-12
    - Eng Units, 3-13
    - Initial Value, 3-12
    - Item, 3-13
    - Max EU, 3-13
    - Max Raw, 3-13
    - Min EU, 3-13
    - Min Raw, 3-13
    - Use Tagname as Item Name, 3-13
  - I/O Discrete details, 3-11
  - I/O Discrete Details, 3-11
    - Access Name, 3-11
    - Initial Value, 3-11
    - Input Conversion, 3-11
    - Item, 3-12
    - Off Msg, 3-11
    - On Msg, 3-11
    - Use Tagname as Item name, 3-12
  - I/O Message Details, 3-14
    - Access Name, 3-14
    - Initial Value, 3-14
    - Item, 3-14
    - Maximum Length, 3-14
  - Inserting into a script, 5-5
  - Local tagname use count, 3-5
  - Logging Events, 3-8
  - Max EU, 3-39
  - Memory Analog Details, 3-9
  - Memory Discrete Details, 3-9
    - Initial Value, 3-9
    - Off Msg, 3-9
    - On Msg, 3-9
  - Memory Message Details, 3-10
    - Initial Value, 3-11
    - Maximum Length, 3-11
  - Min EU, 3-39
  - Placeholder Tagnames, 2-15, 2-17
  - Printing details, 3-32
  - Priority, 3-8
  - Read Only, 3-8
  - Read/Write, 3-8
  - Remote tagname use count, 3-5
  - Replacing in a script, 5-6
  - Retentive Parameters, 3-8
  - Retentive Value, 3-8
  - Scaling, 3-39
  - Selecting a .field, 3-19, 3-43
  - Substituting, 3-37
  - SuperTag Types, 3-7
  - Updating use counts, 3-34
  - Use counts, 3-5
  - Using dashes, 3-6
  - Valid characters, 3-6, 3-7
- Tan(), 5-40

- Text Box Control Wizard, 2-56, 2-57
  - Text Color Tool, 1-15
  - Text Object Tool, 1-16, 2-46
  - Text Objects, 2-2
    - Bold, 1-14
    - Centered, 1-15
    - Changing the text, 2-48, 2-49
    - Creating, 2-46
    - Displaying numeric values, 2-45, 2-46
    - Editing, 1-27
    - Enlarging the font size, 1-15
    - Formatting, 2-45
    - Italics, 1-14
    - Justification, 2-45
    - Left Justified, 1-15
    - Reducing the font size, 1-15
    - Replacing a portion, 2-49
    - Right Justified, 1-15
    - Selecting
      - Font, 2-47
      - Font Point Size, 2-48
    - Selecting a font, 1-14
    - Selecting Color, 1-15
    - Underlining, 1-15
  - Text Tools
    - Bold, 1-14
    - Centered, 1-15
    - Enlarge Font, 1-15
    - Font, 1-14, 2-47
    - Italic, 1-14
    - Left Justified, 1-15
    - Reduce Font, 1-15
    - Right Justified, 1-15
    - Text Color, 1-15
    - Underline, 1-15
  - Text(), 5-39
  - Tick Interval, 2-82
  - Time Zones, 2-101
  - Title Bar text, 2-5
  - Toggle Pushbutton, 4-13
  - Toolbar Tools
    - 3-D Button, 1-16
    - Align Bottom, 1-17, 2-33
    - Align Center, 1-17, 2-30
    - Align Centerpoints, 1-17, 2-33
    - Align Left, 1-17, 2-29
    - Align Middle, 2-32
    - Align Middles, 1-17
    - Align Right, 1-17, 2-31
    - Align Top, 1-17, 2-31
    - Application Explorer, 1-16
    - Bitmap, 1-16, 2-40, 2-42, 2-43
    - Bold, 1-14
    - Break Cell, 1-18, 2-38
    - Break Symbol, 1-18, 2-38
    - Bring to Front, 1-17, 2-34
    - Centered, 1-15
    - Close Window, 1-13, 2-13
    - Copy Object, 1-13, 2-22
    - Cut Object, 1-13, 2-22
    - Diagonal Line, 1-15
    - Duplicate Object, 1-13, 2-21
    - Ellipse, 1-15
    - Enlarge Font, 1-15
    - Fill Color, 1-15
    - Flip Horizontal, 1-18, 2-36
    - Flip Vertical, 1-18, 2-36
    - Font, 1-14, 2-47
    - Full Screen Mode, 1-16
    - Hide All Toolbars, 1-16
    - Horizontal/Vertical Line, 1-15
    - Italic, 1-14
    - Left Justified, 1-15
    - Line Color, 1-15
    - Make Cell, 1-18, 2-38
    - Make Symbol, 1-18, 2-38
    - New Window, 1-13
    - Open Window, 1-13, 2-12
    - Paste Object, 1-14, 2-23
    - Polygon, 1-16
    - Polyline, 1-16
    - Print, 1-14
    - Real time Trend, 1-16, 7-2
    - Rectangle, 1-15
    - Redo, 1-14, 2-20
    - Reduce Font, 1-15
    - Reshape Object, 1-18, 2-26
    - Right Justified, 1-15
    - Rotate Clockwise, 1-18, 2-36
    - Rotate CounterClockwise, 1-18, 2-36
    - Rounded Rectangle, 1-15
    - Ruler, 1-17
    - Save All Windows, 1-13
    - Save Window, 1-13, 2-12
    - Selector, 1-15
    - Send to Back, 1-17, 2-34
    - Snap to Grid, 1-17, 2-39
    - Space Horizontal, 1-18, 2-35
    - Space Vertical, 1-18, 2-35
    - Text Color, 1-15
    - Text Object, 1-16, 2-46
    - Transparent Color, 1-15
    - Underline, 1-15
    - Undo, 1-14, 2-19, 2-20
    - Window Background Color, 1-15
  - Topic Name, 8-5
  - Touch Action Scripts Editor dialog box, 5-12
  - Touch Links. *see* Animation Links
  - Touch Pushbutton Action Scripts, 5-12
  - Transparent Bitmaps, 2-43
  - Transparent Color Tool, 1-15, 2-44
  - Trend Objects, 2-3
  - Trending
    - Historical, 7-1
    - Real time Trend Tool, 1-16, 7-2
    - Real-time, 7-1
  - Trunc(), 5-40
- ## U
- Underline Tool, 1-15
  - Undo
    - Levels, 1-14, 2-7
    - Object Edits, 2-20

- Stacks, 2-7
- Undo Tool, 1-14, 2-19, 2-20
- Universal Coordinated Time, 2-101
- Updating
  - All trends "Fast", 2-82
  - Time Variables, 2-82
  - Use Counts, 3-5, 3-34
- Use Application resolution, 2-95
- Use old SendKeys, 2-82
- Use Tagname as Item Name, 3-12, 3-13
- Using
  - ActiveX Control Event Parameters, 2-74
  - ActiveX Control Methods, 2-70
  - I/OStatus Topic Name, 8-13
  - I/OStatus Topic Name in Excel, 8-16
  - InTouch QuickScript Editor, 5-3
  - Local Variables, 5-21
  - Security Internal Tagnames, 2-87
  - WindowMaker Help, 1-34

## V

- Validating Scripts, 5-7
- Value Deadband, 3-16
- Value Display Links
  - Analog, 4-29, 4-30
  - Discrete, 4-29
  - String, 4-29, 4-30
- Value Time Quality, 8-2
- Vertical Percent Fill Links, 4-23
- Vertical Slider Links, 4-11
- View Toolbar, 1-16, 2-18
  - Application Explorer Tool, 1-16
  - Full Screen Mode Tool, 1-16
  - Hide All Toolbars Tool, 1-16
  - Ruler Tool, 1-17
  - Snap to Grid Tool, 1-17, 2-39
- Viewing the Cross Reference Search Results, 3-25
- Visibility Links, 4-25
- VTQ, 8-2

## W

- wcAddItem(), 5-44
- wcClear(), 5-44
- wcDeleteItem(), 5-44
- wcDeleteSelection(), 5-44
- wcErrorMessage, 5-50
- wcErrorMessage(), 5-44
- wcFindItem(), 5-44
- wcGetItem(), 5-44
- wcGetItemData(), 5-44
- wcInsertItem(), 5-44
- wcLoadList(), 5-44
- wcLoadText(), 5-44
- wcSaveList(), 5-44
- wcSaveText(), 5-44
- wcSetItemData(), 5-44
- While False Condition Scripts, 5-14, 5-15
- While Running Application Scripts, 5-9
- While Showing Window Scripts, 5-10
- While True Condition Scripts, 5-14, 5-15

- Window Background Color Tool, 1-15
- Window Configuration dialog box, 2-83
- Window Properties dialog box, 2-8
- Window Scripts, 5-2
  - On Hide, 5-10
  - On Show, 5-10
  - While Showing, 5-10
- WindowMaker
  - Closing on transfer to WindowViewer, 2-6
  - Color Palette, 1-21, 2-44
  - GUI, 1-2
  - Printout dialog box, 1-14, 5-37
  - Program Elements, 1-1
  - Properties, 2-5
  - Properties dialog box, 2-5
  - Ruler, 1-19
  - Status Bar, 1-20
- WindowMaker Toolbars
  - Arrange, 1-11, 1-17, 2-18, 2-29
  - Changing the size of a floating Toolbar, 1-12
  - Docking, 1-11
  - Draw Object, 2-18
  - Draw Object Toolbar, 1-15
  - Floating, 1-11
  - Floating a docked Toolbar, 1-12
  - Format, 1-14, 2-45
  - General, 1-13, 2-8, 2-12, 2-18
  - Hiding all, 1-13
  - Show/Hide a docked Toolbar, 1-11
  - Show/Hide a floating Toolbar, 1-12
  - View, 1-16, 2-18
  - Wizards/ActiveX, 1-14, 2-18
- Windows
  - Background Color, 2-9
  - Closing, 2-13
  - Creating a new window, 2-8
  - Deleting, 2-13
  - Dimensions, 2-10
  - Duplicating, 2-13
  - Exporting, 2-14
  - Frame Style, 2-10
  - Height, 2-10
  - Importing, 2-16, 2-17
  - Importing Scripts, 2-17
  - Miscellaneous comments, 2-9
  - Naming, 2-9
  - Opening, 2-12
  - Saving, 2-12
  - Scripts, 2-11
    - Importing, 2-17
  - Title Bar, 2-10
  - Types, 2-9
  - Width, 2-10
  - X Location, 2-10
  - Y Location, 2-10
- Windows Control Wizards, 2-56, 5-44
  - .Caption, 2-61
  - .Enabled, 2-61
  - .ListCount, 2-61
  - .ListIndex, 2-61
  - .NewIndex, 2-62
  - .ReadOnly, 2-62

- .TopIndex, 2-62
- .Value, 2-62
- .Visible, 2-62
- Check Box, 2-59
- Check Boxes, 2-56
- Combo Box, 2-58
- Combo Boxes, 2-56
- Configuring, 2-60
- Control Name, 2-56, 2-61
- Drop Down Combo Box, 2-59
- Drop Down List Combo Box, 2-59
- Getting & Setting Properties, 2-61
- List Box, 2-58
- List Boxes, 2-56
- Option Buttons, 2-56
- Properties, 2-61
- Radio (Option) Buttons, 2-60
- Radio Buttons, 2-56
- Simple Combo Box, 2-59
- Text Box, 2-57
- Usability Guidelines, 2-57
- Windows Controls Script Functions
  - Error Messages, 5-50
  - wcAddItem(), 5-44
  - wcClear(), 5-44
  - wcDeleteItem(), 5-44
  - wcDeleteSelection(), 5-44
  - wcErrorMessage(), 5-44
  - wcFindItem(), 5-44
  - wcGetItem(), 5-44
  - wcInsertItem(), 5-44
  - wcLoadList(), 5-44
  - wcLoadText(), 5-44
  - wcSaveList(), 5-44
  - wcSaveText(), 5-44
  - wcSetItemData(), 5-44
- Windows NT
  - DDE resources, 8-3
  - Setting up a DDE Share, 8-3
- Windows Scripts
  - On Hide, 2-11
  - On Show, 2-11
  - While Showing, 2-11
- Windows Scripts Editor dialog box, 5-10
- Windows to Export dialog box, 2-14
- Windows to Print Dialog Box, 3-33
- Windows to Show when touched
  - dialog box, 4-16
- WindowViewer
  - Cannot start as NT Service, 2-97
  - Closing all open windows on transfer to WindowMaker, 2-81
  - Closing on transfer to WindowMaker, 2-81
  - Customizing, 2-80
  - Disabling transfer to WindowMaker, 2-84
  - Keeping maximized, 2-85
  - Logic Menu, 2-84
  - Properties, 2-80
  - Selecting
    - Home Windows, 2-85
  - Showing/Hiding Menu bar, 2-84
  - Starting as an icon, 2-81
- WindowViewer Properties dialog box, 2-80
- Wizard Selection dialog box, 1-14, 2-53, 2-66
- Wizard/ActiveX Installation dialog box, 2-52
- Wizards, 2-4
  - Adding to the toolbar, 2-54
  - Installing, 2-52
  - Pasting in a window, 2-53
  - Removing, 2-52
  - Removing from the toolbar, 2-54
- Wizards/ActiveX Toolbar, 1-14, 2-18
  - Wizard Tool, 1-14, 2-53, 2-66, 2-67
- Wonderware Logger, 2-6, 2-81
  - Starting automatically, 2-6, 2-81
- Wonderware Service User dialog box, 2-99
- Wonderware SuiteLink Communication Protocol, 8-1, 8-3
- Working with
  - ActiveX Controls, 2-64
  - Floating/Docking Toolbars, 1-11
  - Graphic Objects, 2-18
  - Images and Bitmaps, 2-40
  - Lines and Outlines, 2-51
  - Text Objects, 2-45
  - WindowMaker Windows, 2-8
  - Wizards, 2-52
- WWControl(), 5-45
- WWExecute(), 5-45
- WWPoke(), 5-45
- WWRequest(), 5-45