

GraphWorX

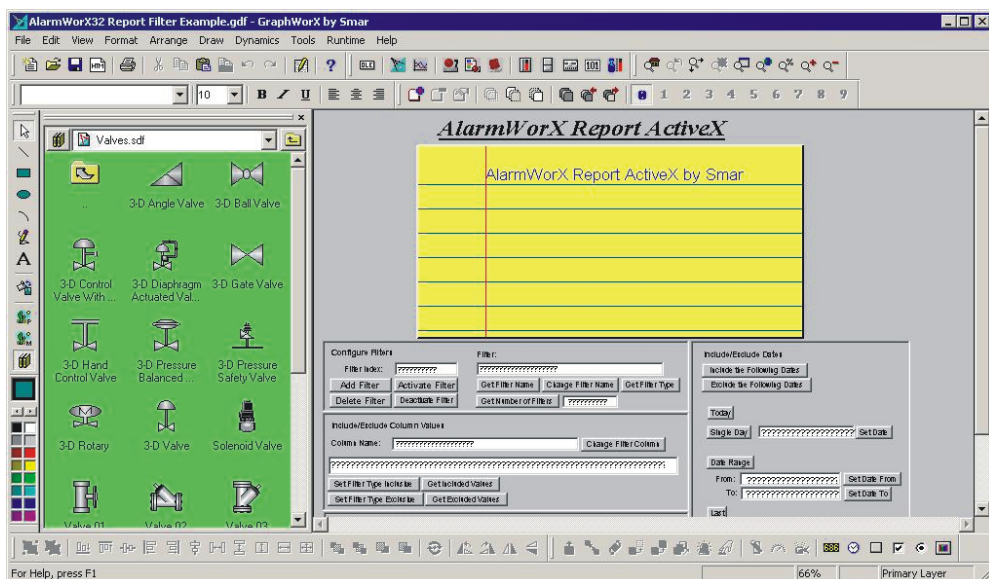
smar
FIRST IN FIELDBUS

JUN / 04
GraphWorX
VERSION 7.1



USER'S GUIDE

GraphWorX



smar



web: www.smar.com

**Specifications and information are subject to change without notice.
For the latest updates, please visit the SMAR website above.**

BRAZIL

Smar Equipamentos Ind. Ltda.
Rua Dr. Antonio Furlan Jr., 1028
Sertãozinho SP 14170-480
Tel.: +55 16 3946-3510
Fax: +55 16 3946-3554
e-mail: smarinfo@smar.com

GERMANY

Smar GmbH
Rheingaustrasse 9
55545 Bad Kreuznach
Germany
Tel.: + 49 671-794680
Fax: + 49 671-7946829
e-mail: infoservice@smar.de

USA

Smar International Corporation
6001 Stonington Street, Suite 100
Houston, TX 77040
Tel.: +1 713 849-2021
Fax: +1 713 849-2022
e-mail: sales@smar.com

ARGENTINA

Smar Argentina
Soldado de La Independencia, 1259
(1429) Capital Federal – Argentina
Telefax: 00 (5411) 4776 -1300 / 3131
e-mail: smarinfo@smarperifericos.com

MEXICO

Smar México
Cerro de las Campanas #3 desp 119
Col. San Andrés Atenco
Tlalnepantla Edo. Del Méx - C.P. 54040
Tel.: +53 78 46 00 al 02
Fax: +53 78 46 03
e-mail: ventas@smar.com

CHINA

Smar China Corp.
3 Baishiqiao Road, Suite 30233
Beijing 100873, P.R.C.
Tel.: +86 10 6849-8643
Fax: +86-10-6894-0898
e-mail: info@smar.com.cn

SINGAPORE

Smar Singapore Pte. Ltd.
315 Outram Road
#06-07, Tan Boon Liat Building
Singapore 169074
Tel.: +65 6324-0182
Fax: +65 6324-0183
e-mail: info@smar.com.sg

FRANCE

Smar France S. A. R. L.
42, rue du Pavé des Gardes
F-92370 Chaville
Tel.: +33 1 41 15-0220
Fax: +33 1 41 15-0219
e-mail: smar.am@wanadoo.fr

Smar Research Corporation

4250 Veterans Memorial Hwy.
Suite 156
Holbrook , NY 11741
Tel: +1-631-737-3111
Fax: +1-631-737-3892
e-mail: sales@smarresearch.com

Index

1. GETTING STARTED.....	1.1
Introduction to GraphWorX.....	1-1
New Features.....	1-1
Additional Features.....	1-2
VBA Smart Wizard Symbols.....	1-3
Starting GraphworX.....	1-4
GraphworX Screen.....	1-4
GraphworX Work Area.....	1-5
Definitions.....	1-5
Status Bar.....	1-5
Toolbars.....	1-5
Main Toolbar.....	1-5
Dynamics Toolbar.....	1-6
ActiveX Toolbar.....	1-6
Draw Toolbar.....	1-6
Arrange Toolbar.....	1-6
Text Style Toolbar.....	1-7
Layers Toolbar.....	1-7
Dockable Symbol Toolbar.....	1-7
Viewing the Dockable Symbol Toolbar in GraphWorX.....	1-8
Features of the Dockable Symbol Toolbar.....	1-9
Symbol Toolbar Menu.....	1-9
Category Locks.....	1-10
Implementation Specifics.....	1-10
Color Palette.....	1-11
Other Features of the Color Palette Tool.....	1-11
Using the Mouse and Keyboard.....	1-12
Mouse Functions.....	1-12
Drag-and-Drop.....	1-13
Keyboard Shortcuts.....	1-13
2. SYSTEM CONFIGURATION.....	2-1
Introduction.....	2-1
Command Line Options.....	2-1
Customizing the Launch of GraphWorX and Windows.....	2-1
Command Line Examples.....	2-2
Application Preferences and Display Properties.....	2-2
Application Preferences.....	2-2
Display Properties.....	2-3
General Parameters.....	2-4
Window Options.....	2-5
Runtime.....	2-6
Runtime Window Properties Mode.....	2-9
Runtime Advanced Settings.....	2-9
Grid.....	2-14
Script Editor.....	2-14
Load Tabs.....	2-15
ToolTips.....	2-16
Compatibility.....	2-17
Visual Basic for Applications.....	2-18

Copying and Resetting Preferences and Properties.....	2-20
Language-Aliasing Support	2-21
Sample Language Configuration	2-21
Creating Language-Aliased Strings.....	2-24
Global Aliasing Support.....	2-29
3. MANAGING DISPLAY FILES	3-1
Introduction	3-1
File Menu.....	3-1
GraphWorX File Name Extensions.....	3-1
Working with GraphWorX Display Files.....	3-2
New	3-2
Open.....	3-2
Save As	3-2
Save	3-3
Save As Previous Version	3-3
Printing GraphWorX Displays.....	3-9
Print Setup.....	3-10
Summary Information	3-10
Exiting GraphWorX.....	3-12
Templates.....	3-12
Apply Template	3-12
Remove Applied Template	3-12
Edit Applied Template	3-13
Exit Edit Applied Template	3-13
Update Template Displays	3-13
Insert Template Objects	3-13
Layers.....	3-13
Configuring Layers	3-14
Layers Toolbar	3-18
Editing Tools.....	3-19
Security for Layers	3-20
Using Layers in Runtime Mode	3-21
Additional VBA Events for Layering	3-22
4. CREATING AND MODIFYING OBJECTS	4-1
Introduction	4-1
Configuration Mode	4-1
Display Configuration Mode Password	4-1
Arrow Key Resize	4-2
TAB Key Object Selection	4-2
ALT Key Group Selection.....	4-2
CTRL+SHIFT Drag.....	4-2
Template Activity	4-2
Find and Replace Features.....	4-2
Customize Toolbars	4-3
Draw Functions.....	4-3
Selector Tool.....	4-4
The Property Inspector	4-4
Checking and Altering the Parameters of an Object	4-5
Gradient Fill.....	4-6
Lines.....	4-10
Segmented Lines	4-11
Rectangles	4-11
Ellipses	4-12
Text	4-13

Arcs	4-15
Import Functions	4-17
Images.....	4-17
Metafiles.....	4-21
Custom Colors.....	4-21
Export Functions.....	4-21
GraphWorX Symbols	4-23
Stand-alone Symbol Library	4-24
Dockable Symbol Toolbar	4-24
Registry Presettings	4-25
Formatting Objects	4-26
Background Color.....	4-26
Color Palette Menu.....	4-26
Eyedropper.....	4-27
Reset Default Colors	4-27
Fill Color	4-27
Line Color	4-28
Line Style.....	4-28
Line Width	4-28
Toggle Fill.....	4-28
Toggle Freeze	4-28
Font	4-29
Arranging Objects	4-29
Group Into Symbol	4-29
Ungroup.....	4-30
Aligning Objects	4-30
Bring to Front.....	4-31
Send to Back.....	4-31
Bring Forward.....	4-31
Send Backward	4-31
Rotation	4-31
Free Rotation of Static Objects	4-31
Rotate Left and Right	4-32
Flip Horizontal/Vertical	4-33
5. DISPLAY EDIT FUNCTIONS	5-1
Introduction to Editing Functions	5-1
Undo/Redo.....	5-1
Basic Editing.....	5-2
Copy.....	5-2
Cut.....	5-2
Paste	5-3
Paste Special	5-3
Duplicate	5-4
Select All	5-4
Find, Report, and Replace.....	5-5
User Interface for Find and Replace	5-8
Wildcards	5-10
Insert Object	5-10
Update Shared Objects	5-12
User Interface for Update Shared Objects	5-12
Where and How to Use Shared Objects	5-13

6. VIEWING FILES	6-1
View Functions	6-1
The Home View	6-1
Zoom Functions	6-2
Zoom	6-2
Custom Zoom	6-2
Unzoom	6-2
Box Zoom	6-2
Zoom Selection	6-3
Fit to Window	6-3
Show Whole Display	6-3
Decluttering Zoom	6-3
Mouse Zoom	6-4
Summary Information	6-4
Viewing Object Count Statistics	6-5
Hide Layers	6-7
Toolbars	6-8
Viewing Current Coordinates on the Status Bar	6-9
Scrolling	6-9
Horizontal Scroll Bar and Vertical Scroll Bar	6-9
Grid	6-9
Runtime Window Properties Mode	6-10
Properties Window	6-10
Select Language	6-11
7. DATA CONNECTIONS	7-1
Introduction to Data Connections	7-1
Tags	7-1
Tag Browser	7-2
Description of User Interface of the Tag Browser	7-2
Expression Editor	7-2
Writing Expressions	7-3
Strings in Expressions	7-4
Point Extension Syntax	7-5
OPC Tags	7-5
Aliases	7-6
Variables	7-7
Arithmetic	7-8
Relational	7-8
Logical	7-9
Bitwise	7-10
Functions	7-11
Local GraphWorX Variables	7-16
Edit Local Variables Parameters	7-16
Constant Values	7-17
Aliasing Data Connections	7-17
Overview of Aliasing	7-17
Object-Level Aliasing	7-17
Editing Aliases	7-18
Runtime Aliasing	7-19
Second-Level Aliasing	7-22
Editing Data Source Connections	7-22

8. DYNAMIC CONNECTIONS.....	8-1
The Data Source.....	8-1
Action Dynamics	8-1
Size Connection	8-1
Location/Slider.....	8-4
Rotation/Dial.....	8-6
Hide/Disable	8-8
Digital Color.....	8-9
Analog Color.....	8-11
Flash.....	8-13
Pick Actions	8-15
Parameters for Pick Action.....	8-16
Selector Dynamics.....	8-30
Digital Selector	8-30
Analog Selector	8-32
Animator	8-33
Intrinsic Dynamics.....	8-35
Process Points and Data Entries.....	8-36
State Fields	8-38
Time/Date.....	8-40
Push Button.....	8-40
Check Box	8-41
Radio Button.....	8-43
Display Button Wizard	8-44
Customizing	8-47
Custom Configuration.....	8-47
Custom Command Execution.....	8-49
9. TOOLS.....	9-1
Tools Menu.....	9-1
Function Keys.....	9-1
Set Working Directory.....	9-3
Security.....	9-3
Local Alias File Editor	9-4
Publishing to HTML	9-5
Basic Steps in Publishing GraphWorX Displays	9-5
Delivering the Necessary Web Components to the Client	9-7
Multiple Display Support.....	9-7
Embedded ActiveX Control Support.....	9-8
Using the Web Publishing Wizard	9-8
Starting the Web Publishing Wizard.....	9-8
Exporting a Display File Locally	9-10
Publishing a Display File to a Web Server	9-12
Publishing Customization Options.....	9-14
Viewing Locally Exported HTML Files.....	9-17
Viewing Published HTML Files.....	9-17
Configuring for Windows CE in ProcessView	9-18
Downloading ProcessView Configuration Files to Your Pocket PC.....	9-19
Setting up the Download	9-20
Configuring the Desktop.....	9-20
File Download Application Configuration.....	9-21
Downloading the File to the CE Device.....	9-23
Scripting Engine and Script Editor for VBScript and JScript.....	9-25
Displaying the Script Editor	9-26
Event Mode	9-27

Object Mode	9-28
Changing the Script Name	9-29
Basic Text Operation	9-29
Text Search and Replace	9-30
Importing and Exporting Scripts	9-30
Print Scripts	9-31
Preferences and Settings	9-32
10. RUNTIME ENVIRONMENT	10-1
Starting Runtime	10-1
Menu Bar Option	10-1
Shortcut on Desktop	10-1
Command Line Option	10-1
Runtime File Menu	10-2
File Functions	10-2
File - Open	10-2
Printing the Screen	10-2
Print Preview	10-2
File - Print Setup	10-3
Runtime View Menu	10-3
Data Statistics	10-3
Display History	10-4
Runtime Security	10-4
Runtime Data Entry	10-4
TraceWorX Support	10-4
OLE Express Compatibility	10-5
11. VBA WIZARDS	11-1
Introduction to VBA Wizards	11-1
How VBA Wizard Works	11-1
Design Mode	11-1
Runtime Mode	11-1
Rules for VBA Wizard	11-2
VBA Wizard Creation Tool	11-2
Macro or Script Name	11-3
Module	11-4
Parameters	11-4
Insert Also a Form, Please	11-4
Form	11-4
More Information About VBA Wizards	11-4
Design Mode VBA Wizard	11-4
Runtime and Design Mode VBA Wizard	11-5
Runtime Mode VBA Wizard	11-5
Other Sources of Information	11-6
12. GRAPHWORX ACTIVEX CONTROL	12-1
Inserting the GraphWorX ActiveX	12-1
GraphWorX ActiveX Toolbar	12-2
Customizing the ActiveX Toolbar	12-2
OLE ActiveX	12-2
Graphics ActiveX	12-3
Trend ActiveX	12-3
Alarm ActiveX	12-4
Gauge ActiveX	12-4
Switch ActiveX	12-4
Slider ActiveX	12-5

Numeric ActiveX.....	12-5
Vessel ActiveX	12-5
National Instruments ActiveX	12-6
Configuring the GraphWorX ActiveX	12-6
13. OLE AUTOMATION REFERENCE.....	13-1
Introduction	13-1
Registration at GenRegistrar in Design Mode	13-1
Object Hierarchy	13-2
Automation Programming in C++ and Visual Basic: Understanding Inheritance.....	13-3
Events.....	13-4
Properties and Methods.....	13-8
Gwxview	13-9
GwxDisplay	13-12
GwxVisible.....	13-38
GwxText	13-43
GwxRectangle	13-44
GwxArc.....	13-45
GwxSymbol	13-45
GwxOleObject	13-46
GwxEllipse.....	13-47
GwxLine	13-47
GwxBitmap	13-47
GwxMetafile.....	13-47
GwxButton	13-47
GwxDynamic	13-48
GwxDigitalSelector	13-50
GwxDigitalSelectorInfo	13-50
GwxDigitalColor.....	13-50
GwxDigitalColorInfo.....	13-51
GwxAnalogSelector	13-51
GwxAnalogColor	13-52
GwxAnimator.....	13-52
GwxSize	13-52
GwxLocation.....	13-52
GwxRotation.....	13-53
GwxFlash	13-53
GwxHide.....	13-53
GwxPick	13-54
GwxProcessPoint	13-55
GwxTimedate	13-56
GwxPoint	13-56
14. GRAPHWORX VBA PROJECT.....	14-1
GwxTools Module	14-1
ThisDisplay Module	14-1
Event Handling with the <i>ThisDisplay</i> Module.....	14-2
How to Control GraphWorX Automation Using VBA	14-3
How to Access GraphWorX Native Display Objects	14-4
Object Names Must Be Unique	14-7
Using VBA to Connect With Other Applications	14-8
Other Sources of Information.....	14-11

15. GRAPHWORX TRANSLATOR UTILITY	15-1
Introduction to the GraphWorX Translation Utility	15-1
Converting Displays.....	15-1
Native 16-bit GraphWorX displays	15-2
Third-Party Displays.....	15-3
Translator Controls.....	15-4
Main Window.....	15-4
Browsing Source Files.....	15-4
Browsing the Destination Directory	15-5
Source File List Control	15-5
List View	15-6
Report View	15-6
List View and List Box	15-7
List View and GraphWorX Preview	15-7
Option Controls.....	15-7
General.....	15-8
Display.....	15-8
Window.....	15-8
Tag.....	15-9
Script Language Reference	15-9
Syntax	15-9
Keyword Keys	15-10
Window Keys.....	15-10
Visible Keys.....	15-11
Dynamic Keys	15-14
16. FIX TO GWX TRANSLATION UTILITY	16-1
Introduction	16-1
What Are .jdf Files?	16-1
Using the Translator	16-1
Limitations of Translation.....	16-2
Translator Limitations	16-2
Intellution "Web Export" Limitations (v1.5)	16-2
FIX Web Server Conversion Utility	16-3
17. WONDERWARE TO GWX TRANSLATOR	17-1
Introduction	17-1
Important Note.....	17-1
InTouch To GraphWorX Translator	17-1
Conversion Specifics	17-2
Supported Conversion Types.....	17-2
Functional Limitations	17-3
What Will Not Convert?.....	17-3
Translation Impurities	17-3
Win-XML Exporter	17-4
Installation Instructions.....	17-4
18. GRAPHWORX VIEWER ACTIVEX.....	18-1
Introduction	18-1
Inserting the GraphWorX Viewer ActiveX.....	18-1
Configuring the GraphWorX Viewer ActiveX	18-1
GraphWorX Viewer ActiveX Parameters	18-2

Section 1

Getting Started

Introduction to GraphWorX

GraphWorX is a human-machine interface (HMI) software package for process control. GraphWorX is a fully compliant OPC client featuring ActiveX™ and OLE Automation technologies.

New Features

- Greatly improved performance
- Speed Improvement for pop-ups
- Microsoft VBA 6.3 support
- Global aliasing support
- TraceWorX integration for debugging and diagnostics
- Dockable Symbol Library with OLE Automation
- Dockable VBScript and JScript editors
- Web Publishing Wizard for exporting and publishing displays to HTML
- New Transparent to Displays & Objects (2000 / XP only)
- New Display Thumbnail to File Open Function
- New Publish Multiple Graphics pages to WEBServer
- Process point and text improvements
- Improve Screen Updates for Foreground & Dynamics
- Find and replace support for state fields
- Find and replace support for local aliases
- Simplified access to symbol tags and aliases
- Expression Editor enhancements
- AlarmWorX Indicator ActiveX added to GraphWorX toolbar
- VBScript examples in Symbol Library
- Integrate New Database Support into GraphWorX
- Language Alias Browser
- Global Alias Browser
- Expanded image file import and export functions
- Image compression
- Improved runtime zoom and pan
- Automation improvements
- Printing enhancements
- Save as previous version support
- Global Aliasing in Configuration & Runtime Dialogs
- VBA runtime mouse and keyboard events
- Digital selector, analog selector, and animator enhancements
- Pick action enhancements for layers
- Object selection enhancements
- Event log enhancements for logging actions and data
- Transparency and translucency
- Zoom limits for box zoom
- OLEExpress 6.0 compatibility
- Statistics runtime data statistics viewer
- Multiply custom command support
- Security enhancements
- Free rotation of objects
- Edit data source connection capability

Additional Features

GraphWorX includes the following features:

- International language-switching support.
- ActiveSync download to Windows CE and Pocket PCs.
- Pocket PC development mode; support for Compaq, Casio, and HP.
- Real-time thread priority capability for data thread and timer thread.
- Function key and keyboard assignment capability.
- User-defined pop-up menus.
- OPC request types.
- Range support for the Number Pad.
- Numerous user interface enhancements to improve ease-of-use.
- Key disable capability through the Secure Desktop Configurator.
- Universal OLE for Process Control (OPC) connectivity.
- True 32-bit design.
- A fully compliant OPC client.
- Powerful display-creation tools.
- A complete set of drawing and animation tools in an object-oriented environment. Displays can be scaleable (display automatically resizes when you stretch the window) or fixed-scale (for pixel-perfect displays no matter what size the window is).
- Containment of ActiveX controls and OLE objects.
- Embedding ActiveX controls from Smart or third-party ActiveX controls and OLE objects directly into your displays.
- ActiveX document server. GraphWorX is an ActiveX document server, which means its displays can be run in applications such as Microsoft's Internet Explorer™.
- Comprehensive OLE Automation interface.
- A powerful set of OLE Automation methods and properties for programmatically manipulating GraphWorX displays.
- Fast dynamic animation.
- Capability of 50-millisecond updates.
- GraphWorX ActiveX control. GraphWorX includes an ActiveX control ("GWXview32.OCX") that is capable of running GraphWorX displays. This component has all the runtime capabilities of "GraphWorX32.EXE," and can be conveniently embedded into ActiveX languages, such as Visual Basic or HTML pages.
- Advanced symbol library. GraphWorX includes a utility for loading, storing, and organizing GraphWorX symbols. Simply drag-and-drop symbols to and from the symbol library.
- OPC 1.0 and OPC 2.0 Data Access compliant.
- More than 70 animation types supported.
- Development and configuration for Windows CE.
- Powerful AutoCAD style display layers with clutter/declutter.
- PowerPoint style gradient fills.
- State fields that display readable text representing machine states.
- Ability to rotate text in 90-degree increments.
- Web-based look and feel similar to MS Internet Explorer.
- Professionally drawn 2-D and 3-D gradient symbol libraries.
- Runtime Window Properties Mode for improved WYSIWYG configuration.
- View toolbar for convenient access to zooming and layering commands.
- Powerful advanced polyline editing capabilities.
- Aliasing and advanced editing features.
- Password on display files and user-created custom symbols and symbol wizards.
- Smart ActiveX toolbar that allows you to add or remove any ActiveX control to the ActiveX toolbar.
- Keypad and QWERTY Keyboard support to process point dynamic.
- Customizable toolbars.

VBA Smart Wizard Symbols

- Help Wizard. Provides detailed information on the use of the VBA wizards.
- OPC Data Access Wizard. Sets and retrieves OPC data.
- Microsoft Excel Wizard. Integrates data to and from Excel.
- Report Wizard. Creates simple, useful reports.
- Recipe Wizard. Creates recipes.
- Microsoft Word Wizard. Integrates data into MS Word.
- E-mail and Outlook Wizard. Sends e-mail based on VBA events.
- OPC Calculator Wizard. Connects a four-function calculator to a tag on any display.
- OPC KeyPad Wizard. Connects a QUERTY keypad to a tag on any display.
- Runtime Aliasing Wizard. Changes aliases during runtime mode.
- Analog Chart Wizard. Creates displays with analog OPC tags.
- Digital Chart Wizard. Creates displays with discrete OPC tags.

Starting GraphworX

To start GraphWorX from the Windows **Start** menu, select **Programs > Smar > System302 > ProcessView > GraphWorX > GraphWorX**.

GraphworX Screen

The figure below shows the GraphWorX screen with its basic components, including the Color Palette, Main toolbar, Draw toolbar, Arrange toolbar, Font toolbar, and Dynamics toolbar.

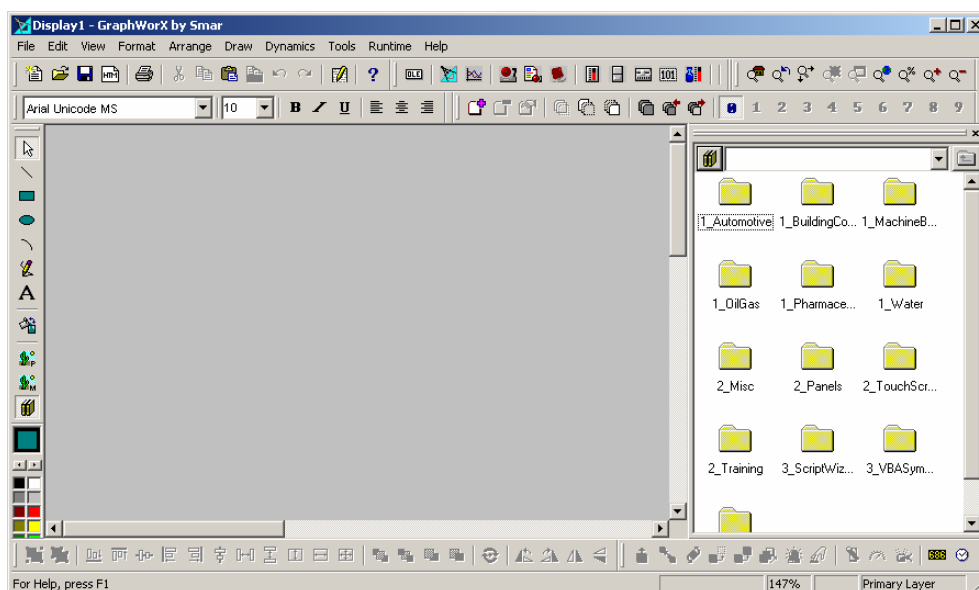


Figure 1.1. GraphWorX Screen

This section describes the features of the GraphWorX screen, as shown above. It also describes the tools used to create displays and make dynamic connections to data values. Refer to your Windows documentation for general procedures to use the Windows Graphical User Interface (GUI) and information about program groups, menus, dialog boxes, as well as file name extensions specific to windows.

GraphworX Work Area

The **Work Area** is where you build your GraphWorX displays. It shows only a portion of the entire work area that can be accessed by the scroll bars or the **Zoom** function.

When the document is saved, information about the current view is saved with the display, so that it can be restored when the display is reloaded at a later time.

Definitions

Display size (or page size): Defines the world coordinate boundaries for the display. The user can specify the display size.

Window dimensions: Refers to the size/location of the GraphWorX mainframe window.

Scaleable display: When the GraphWorX main window is resized, the view is scaled so that all objects currently in the view area remain visible.

Preserve aspect ratio: The ratio of object-sizes in the display is maintained during window resizing.

Fixed-scale display: The view does not scale when the GraphWorX main window is resized. Visible objects are clipped or revealed if the window is sized smaller or larger.

View area: Area of the display that is currently visible.

Home view: View in which the display was last saved.

100% zoom: One logical unit equals one pixel. This is always the case, regardless of whether the display is fixed-scale or scaleable.

Status Bar

The status bar is displayed at the bottom of the GraphWorX window. To display or hide the status bar, select the **Status Bar** command in the **View** menu. The left side of the status bar describes the functions of menu items and toolbar buttons as you navigate through menus and toolbars.

The following items are shown in the status bar:

- Cursor coordinates
- Zoom percentage
- Grid snap on/off
- Display layers

Toolbars

The GraphWorX tools you use to create your displays are grouped by functionality and are available through the menu bar options and the following toolbars:

Main Toolbar

The **Main** toolbar, shown below, is displayed by default across the top of the application window, below the menu bar. The toolbar provides quick mouse access to many tools used in your application. You use the **Main** toolbar for basic file and display maintenance such as creating a new display, opening a file, printing a file, cutting and pasting objects, duplicating objects. It also allows you to undo and redo your last editing action, and to access help on your GraphWorX application. In addition, you can toggle the Script Editor toolbar by clicking the "pencil" button on the **Main** toolbar.



Figure 1.2. Main Toolbar

Dynamics Toolbar

You can use the **Dynamics** toolbar to make dynamic connections to data points in OPC servers. See the **Dynamic Connections** section for a detailed description of tool button functions on the **Dynamics** toolbar.



Figure 1.3. Dynamics Toolbar

Layers Toolbar

Using the **Layers** toolbar, shown below, you can add, remove, or duplicate a layer. You can also edit the layer properties, set the active layer, set the next layer, set the previous layer, hide layers above the current layer and hide layers below the current layer. For more information, please see the *Layers* section below.



Figure 1.8. Layers Toolbar

Dockable Symbol Toolbar

The Symbol Library is available in GraphWorX as a dockable toolbar or a floating window inside display files. It can be freely floating above a GraphWorX application or docked to any side of the GraphWorX display: left, top, right, or bottom, as shown in the figure below. Features of the Symbol Library are fully implemented in the Dockable Symbol Toolbar. You can create, rename, and delete both category (.sdf) files and symbols within the dockable toolbar, as well as unlock categories with passwords and change the look of the symbol icons in the view pane. The category (.sdf) file format of the Dockable Symbol Toolbar is fully compatible with previous versions of the Symbol Library, and vice versa.

GraphWorX uses two different modes for the Symbol Library:

- Standard stand-alone Symbol Library
- Dockable symbol toolbar

To switch between these two modes:

1. Select **Application Preferences** from the **Format** menu.
2. The **Application Preferences** dialog box appears, as shown in the figure below. Select the **Compatibility** tab. Under the **Symbol Library Style** field, you can select **Stand-alone** mode or **Dockable as Symbol Toolbar** mode.

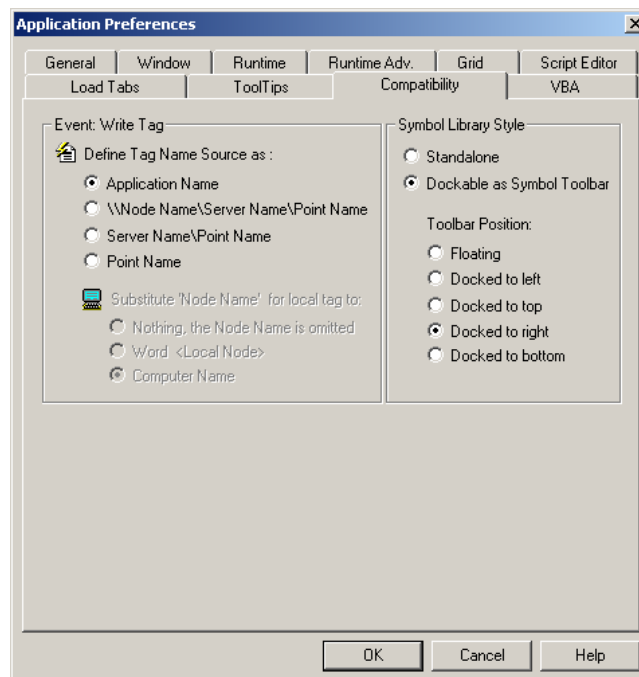


Figure 1.9. Switching Between Stand-alone and Dockable Symbol Modes

If **Stand-alone** mode is active, clicking the **Symbols** button on the **Draw** toolbar launches the **SymbolLibrary.exe**. You can also select **Import > Symbol** from the **Draw** menu, as shown in the figure below.

If the **Dockable as Symbol Toolbar** option is selected, then the Symbol Library is replaced with a Symbol toolbar inside the GraphWorX display, as shown in the figure below. You can also position the symbol toolbar as:

- A floating window
- Docked to left
- Docked to right
- Docked to bottom
- Docked to top

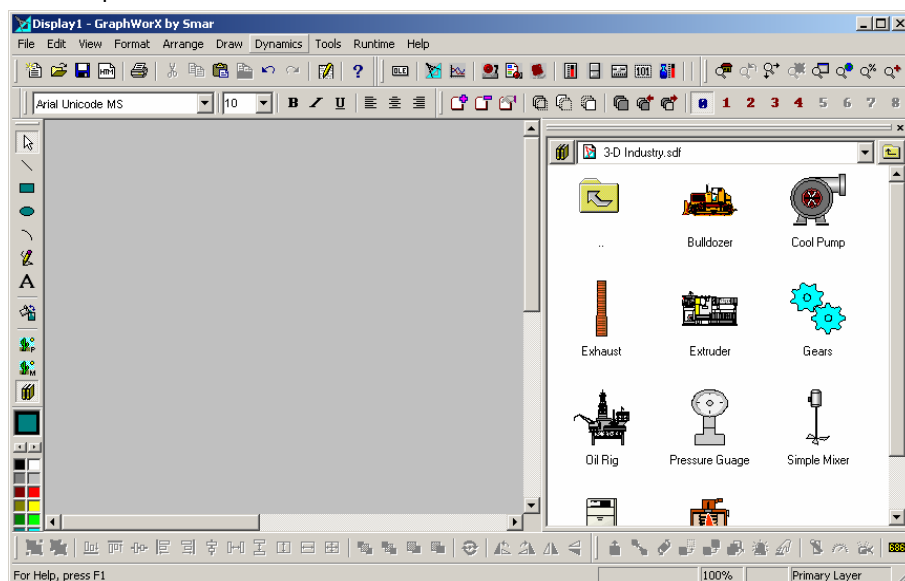


Figure 1.10. Symbol Library Docked Inside a GraphWorX Display

The **Symbols** button on the **Draw** toolbar toggles the visibility of the symbol toolbar (instead of launching the Symbol Library). The **Toggle Symbol Toolbar** command on the **View** menu is enabled, and the **Show Toolbars** dialog (which appears when you select **Toolbars** from the **View** menu) also includes a **Symbols** check box for showing/hiding the symbol toolbar. The **Import > Symbol** command is removed from the **Draw** menu.

Viewing the Dockable Symbol Toolbar in GraphWorX

To view the Dockable Symbol Toolbar inside a GraphWorX display:

1. Select **Toolbars** from the GraphWorX **View** menu, as shown in the figure below.

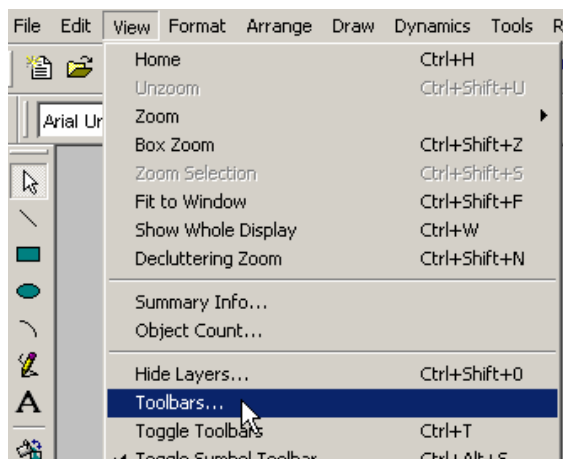


Figure 1.11. Selecting Toolbars in GraphWorX

- In the **Show Toolbars** dialog box, check **Symbols**, as shown in the figure below. Click **OK**.

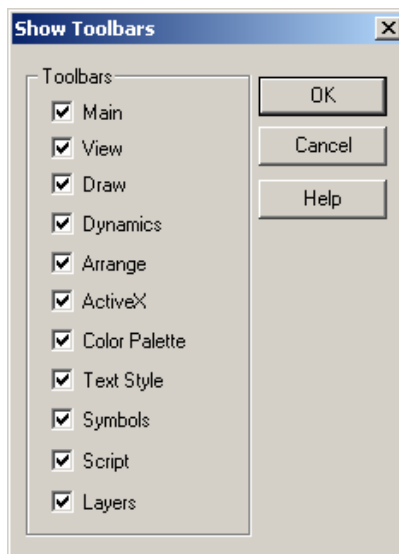


Figure 1.12. Selecting the Symbols Dockable Toolbar

- This displays the Dockable Symbol Toolbar in the GraphWorX display.

Features of the Dockable Symbol Toolbar

The figure below shows the Dockable Symbol Library as a floating window. The drop-down list at the top of the window allows you to select from all available category (.sdf) files. GraphWorX symbols that are contained in the selected category file are displayed in the list control pane, as shown in the figure below. For maintenance and manipulation of symbols and categories, two pop-up menus are available.

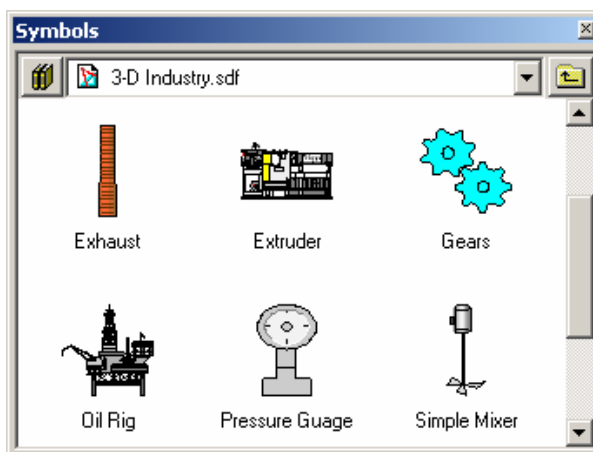


Figure 1.13. Dockable Symbol Library as a Floating Window

Symbol Toolbar Menu

The main pop-up menu for the Docking Symbol Toolbar is displayed when upper-left button is clicked, as shown in the figure below. You can also right-click anywhere inside the window to access this menu. This pop-up menu allows you to add, rename, delete, lock and unlock category files, as well as change the view of the symbol icons in the view pane. You can also cut, copy, paste, delete, and rename the symbols. The main pop-up menu also allows you to specify the root directory for the .sdf files. Previous versions of the Symbol Library maintained a tree control hierarchy of category files; in other words, category files could be placed under different subdirectories. The Dockable Symbol Library lists all category files as upper-level categories (even those included in subdirectories) and tracks where particular a category file came from so that no category files are lost.

Note: Please see the Symbol Library Help for complete information about menu functions.

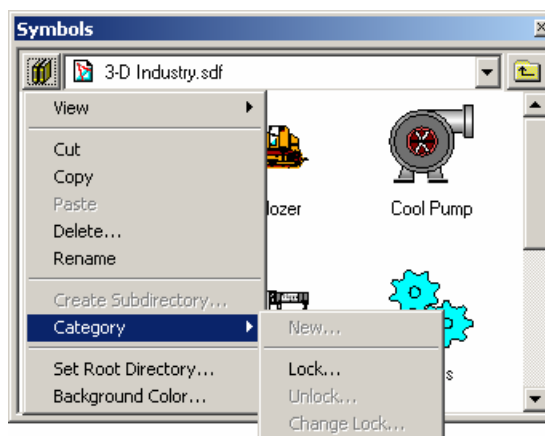


Figure 1.14. Main Pop-up Menu in Dockable Symbol Toolbar

Main Pop-up Menu Commands

Command	Function
View	Specifies the size of the symbol icons in the selected category.
Cut	Removes the selected symbol from the list view.
Copy	Copies the selected symbol.
Paste	Pastes a symbol into the list view.
Delete	Deletes the selected symbol.
Rename	Renames the selected symbol.
Set Root Directory	Specifies the target directory for the symbol category (.sdf) files.
Create Subdirectory	Creates a new subdirectory (folder) in the root directory for the symbol category (.sdf) files.
Category	Adds, renames, deletes, locks, and unlocks symbol category files.
New	Creates a new category file.
Lock	Sets up the read-write password in the Set Category Write Mode Password dialog box. The password must be confirmed. Sets the read-write password to the category, which currently has no locks set, or the read-only password for the category, which is already opened in read-write mode. You cannot set the read-only password without setting the read-write password; this would result in a category file lock that no one would be able to remove (to modify the category file).
Unlock	Opens a locked symbol category. When the category file is initially locked, it is always locked with read-write access denied. (It could also have the read-only lock set). Simply enter the read-write password (or the read-only password if it is configured) to open the file. When the password is entered, it is recognized as either read-write or read-only. If the password is accepted, the category file transitions to the proper state to reflect the change.
Change Lock	Changes the category file password. You must be granted read-write permission to open the category with the read-write password.
Rename	Renames the selected category file.
Delete	Deletes the selected category.
Background Color	Changes the background color of the list view pane.

Category Locks

It is possible to lock any category file included in the drop-down list for writing to and even for reading from the symbol category. If a particular category is locked, it is indicated by different category icon in the list box, as shown in the figure below. Also, if the category file is locked both for reading and writing, a message will appear on the Dockable Symbol Toolbar notifying the user about the lock.

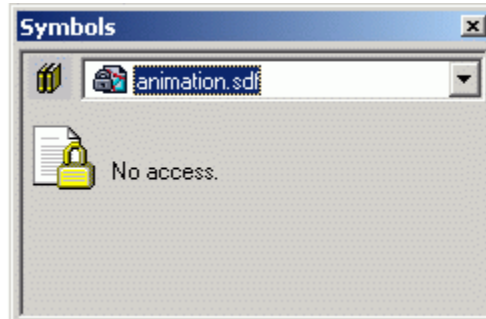


Figure 1.15. Locked Symbol Category File Indicated

Implementation Specifics

Some of Docking Symbol Toolbar serializable information (background color, current category selection, working root directory) is saved in the registry under:

HKCU\Software\ICONICS\SymbolLibrary\Settings

The rest of the toolbar data are saved in the registry under:

HKCU\Software\ICONICS\Gwx\.

IMPORTANT NOTE: Due to saving extra information about the Dockable Symbol Toolbar into the registry, GraphWorX Version 6.1 cannot run on the same machine where GraphWorX Version 7.0 was previously launched unless HKCU\Software\ICONICS\Gwx\General-Summary\Bars is deleted. This allows GraphWorX 6.1 to start with default settings.

Color Palette

A **Color Palette** tool is provided for selecting new colors during configuration mode in the GraphWorX application. This includes fill, border, background and text colors.

The Color Palette, shown below, operates both as a toolbar and/or a dialog box.



Figure 1.16. Color Palette Toolbar

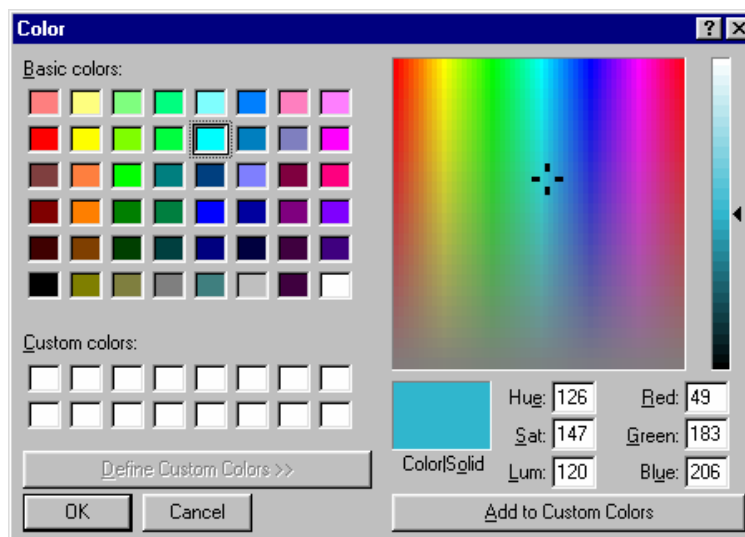


Figure 1.17. Color Palette Dialog Box

The color palette tool consists of two or more rows of selectable colors, a spin button control for accessing additional colors not currently visible, and a current selected color indicator.

Fill color is changed by left-clicking on one of the color boxes.

Line color is changed by right-clicking on one of the color boxes. For text objects, right-clicking on one of the color boxes changes the color of the text.

Double-clicking on a color box is used to select a custom color (via the windows custom color common dialog).

If two or more objects are grouped together, it is possible to change the fill and line color of both objects.

The toolbar can be reoriented while it is floating to change between horizontal and vertical orientations.

Other Features of the Color Palette Tool

Right-clicking on the empty area surrounding the spin button control of the color palette toolbar will bring up the following popup menu.

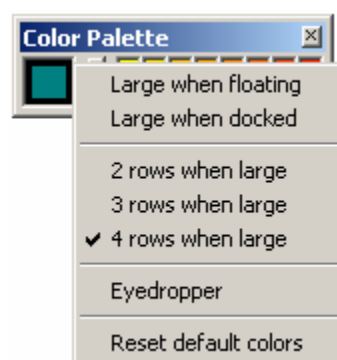


Figure 1.18. Color Palette Tool Menu

This menu can be used to change the size of the color palette (large/small) and to set the number of rows of colors visible at a time (two, three, or four). There is also an option to reset the factory default color definitions of the color palette.

Note: The eyedropper feature can be used to pick up colors from a display to add them to the color palette tool. To use this, choose **Eyedropper** from the popup menu, as shown above (the mouse pointer will change to an eyedropper). Click anywhere in the GraphWorX window over the color you want to pick up. Deposit the color in the color palette by clicking over the desired color box in the rows of available colors.

Using the Mouse and Keyboard

You can use both the mouse and keyboard to control GraphWorX. This section explains how to operate the mouse and lists keyboard shortcuts for GraphWorX.

Mouse Functions

In Windows, you use the mouse to move a pointer on the screen, usually in the shape of an arrow. The pointer shows you where you are on the screen and allows you to point to and select application items.

Function	Description
Left-click	Selects static objects, such as rectangles and ellipses, tools, and menus.
Double-click	Double-clicking an object opens the Property Inspector.
SHIFT + Left-click	Selects multiple objects one by one.
Right-click	Opens the Format menu. Right-clicking on the empty display shows the Format menu.
Drag (left mouse button) and object or an object's handles	Moves, stretches, and resizes objects. Docks and undocks toolbars.
Hold down SHIFT while stretching object	Keeps the size of objects in proportion after they are drawn.
Hold down CTRL while stretching object	Resizes from the center of an object.
Hold with CTRL + SHIFT	Stretches the object in proportion from the center.
Hold down ALT while grid is on	Object will not snap to grid and will turn off snap to grid temporarily.
Hold down SHIFT while moving object if move icon will not work	Will keep object in the same horizontal and vertical plane.
Drag-and-drop	Objects can be dragged and dropped to other instances of GraphWorX and the Symbol Library.
Drag-and-drop from GraphWorX display to desktop	Becomes an icon and can be dragged back as the original object.
CTRL + Drag	Copy and duplicate within the display.
Hold down SHIFT while creating a rectangle or a circle	Forces the object to be a perfect square or circle.
Hold down SHIFT while drawing a line	Line segments are drawn in increments of 45 degrees.
SHIFT + Right-click	To edit text, arc, line, or symbol objects.

Note: If objects are selected, use cursor keys to move by one pixel.

Drag-and-Drop

GraphWorX supports the drag-and-drop of native GraphWorX objects and OLE objects. Static and dynamic GraphWorX objects can be dragged from one instance of GraphWorX into another instance of GraphWorX as native objects. Furthermore, objects can be dragged onto the Windows desktop (as a scrap) and can later be dropped back into GraphWorX as native objects. Objects dropped into GraphWorX that are not in the native GraphWorX format (or are not one of the other clipboard formats supported by GraphWorX) become OLE embedded objects. Other supported clipboard formats include text, bitmaps, and Windows metafiles.

Keyboard Shortcuts

File Functions	
CTRL+N	Opens a new file
CTRL+O	Opens an existing file
CTRL+S	Saves the current file
CTRL+P	Prints the current file

Edit Functions	
CTRL+Z	Undo
CTRL+X	Cut
CTRL+C	Copy
CTRL+V	Paste
CTRL+D	Duplicate
DEL	Delete
CTRL+F	Find

View Functions	
CTRL+H	Home
CTRL+SHIFT+U	Zoom out
CTRL+SHIFT+Z	Box zoom
CTRL+SHIFT+S	Zoom in
CTRL+F	Fit to window
CTRL+W	Show whole display
CTRL+SHIFT+N	Decluttering zoom
CTRL+SHIFT+O	Hides layers
CTRL+T	Allows you to set the visibility of the toolbars
CTRL+B	Shows/hides the status bar
CTRL+L	Toggles the scroll bars
CTRL+SHIFT+G	Shows the grid
CTRL+R	Runtime window properties mode
F4	Opens the Properties window
CTRL+ALT+U	Select Language
CTRL+0	Custom zoom
CTRL+1	Zoom 50%
CTRL+2	Zoom 75%
CTRL+3	Zoom 100%
CTRL+4	Zoom 150%
CTRL+5	Zoom 250%
CTRL+NUMPAD+	Zoom in
CTRL+NUMPAD-	Zoom out
CTRL+M	Toggles configuration and runtime modes

Arrange Functions	
CTRL+G	Group into symbol
CTRL+U	Ungroup symbol
CTRL+SHIFT+PgUp	Brings object to front
CTRL+SHIFT+PgDn	Sends object to back
CTRL+PgUp	Brings object forward
CTRL+PgDn	Sends object backward

Align	
CTRL+ SHIFT+T	Tops
CTRL+ SHIFT+B	Bottoms
CTRL+ SHIFT+M	Middles
CTRL+ SHIFT+L	Lefts
CTRL+ SHIFT+R	Rights
CTRL+ SHIFT+C	Centers

Space Evenly	
CTRL+ SHIFT+A	Across
CTRL+ SHIFT+D	Down

Make Same Size	
CTRL+ SHIFT+H	Height
CTRL+ SHIFT+W	Width
CTRL+SHIFT+O	Both

Move Object	
ARROW LEFT	Moves the object 1 pixel to the left
ARROW RIGHT	Moves the object 1 pixel to the right
ARROW UP	Moves the object 1 pixel up
ARROW DOWN	Moves the object 1 pixel down
CTRL+ARROW LEFT	Moves the object 10 pixels to the left
CTRL+ ARROW RIGHT	Moves the object 10 pixels to the right
CTRL+ ARROW UP	Moves the object 10 pixels up
CTRL+ ARROW DOWN	Moves the object 10 pixels down

Move Border	
SHIFT+ARROW LEFT	Moves the right border 1 pixel to the left
SHIFT+ARROW RIGHT	Moves the right border 1 pixel to the right
SHIFT+ARROW UP	Moves the upper border 1 pixel up
SHIFT+ARROW DOWN	Moves the upper border 1 pixel down
CTRL+ SHIFT+ARROW LEFT	Moves the left border 1 pixel to the left
CTRL+ SHIFT+ARROW RIGHT	Moves the left border 1 pixel to the right
CTRL+ SHIFT+ARROW UP	Moves the bottom border 1 pixel up
CTRL+ SHIFT+ARROW DOWN	Moves the bottom border 1 pixel down

Note:

System Configuration

Introduction

This section explains the different ways of configuring and customizing your system. You can customize your GraphWorX system using:

- Command line options.
- **Application Preferences** dialog box.
- **Display Properties** dialog box.

Command Line Options

You can customize a GraphWorX program item in Windows 95 and Windows NT. This command line feature allows you to start GraphWorX with a specified display and launch GraphWorX directly in runtime. The command line options are listed in the table below.

Command Line Options

Option	Description
Runtime	Launches GraphWorX in Runtime, automatically suppressing splash screen.
NoSplash	No splash screen.
RegServer	Registers application.
UnregServer	Unregisters application.

Customizing the Launch of GraphWorX and Windows

To customize the launch of GraphWorX and Windows, use the following procedure:

1. Right-click **Start** and select **Open**. The **Start** menu appears. Left-clicking **Start** also opens the **Start** menu.
2. Open the **Programs** folder.
3. Open the ProcessView folder from the **Programs** menu.
4. Right-click the GraphWorX icon and select **Properties**. The **GraphWorX Properties** dialog box opens, as shown below.
5. Click the **Shortcut** tab and enter the appropriate Command Line options at the end of the Target field.

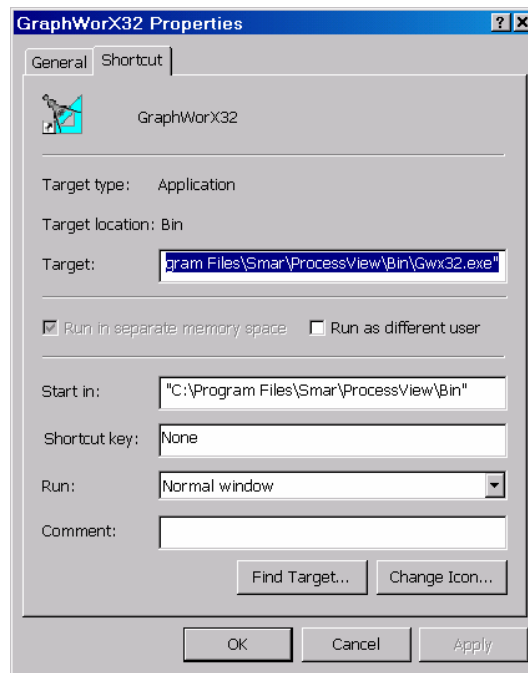


Figure 2.1. Shortcut to GraphWorX Properties Dialog Box

Command Line Examples

In the following Command Line example, GraphWorX starts with the Tanks.gdf display file:

```
C:\ProcessView\GraphWorX.EXE C:\ProcessView\PROJECT1\TANKS.GDF
```

You can also combine two or more of these options. In the following example, GraphWorX starts in Runtime with the Tanks.gdf display loaded and the application window in the middle of your screen:

```
C:\ProcessView\GraphWorX.EXE C:\ProcessView\PROJECT1\TANKS.GDF - Runtime
```

Application Preferences and Display Properties

Application Preferences

The **Application Preferences** dialog box, shown below, allows you to set default operating parameters for your display-editing environment. These parameters apply to each new display created in GraphWorX. Select **Application Preferences** from the **Format** menu. The **Application Preferences** dialog box opens, as shown in the figure below. Here you can set the defaults for the display. Since GraphWorX Preferences apply only to new displays, changes made in this dialog box will not be seen until you select **New** from the **File** menu.

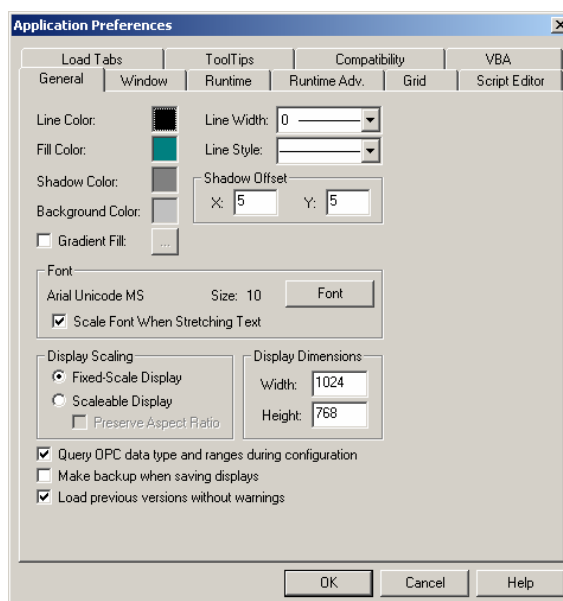


Figure 2.2. Application Preferences Dialog Box

Display Properties

You can also define the properties for the current display by selecting **Display Properties** from the **Format** menu. This opens the **Display Properties** dialog box, shown below. Any changes you make in this dialog box will be immediately reflected in the display.

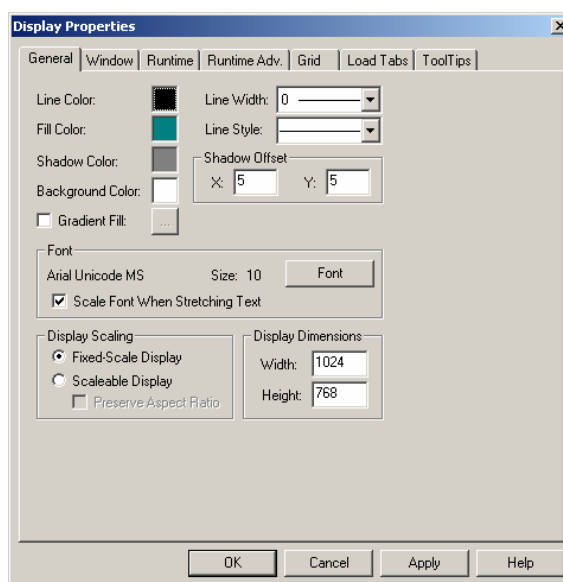


Figure 2.3. Display Properties Dialog Box

As you can see, the **Display Properties** dialog box is very similar to the **Application Preferences** dialog box. Both dialog boxes contain following tabs:

- General Parameters
- Window Options
- Runtime
- Grid
- Script Editor
- Load Tabs
- ToolTips
- Compatibility
- VBA

General Parameters

The **General** tab of the **Application Preferences** and **Display Properties** dialog boxes, shown below, allows you to adjust the settings for the GraphWorX display.

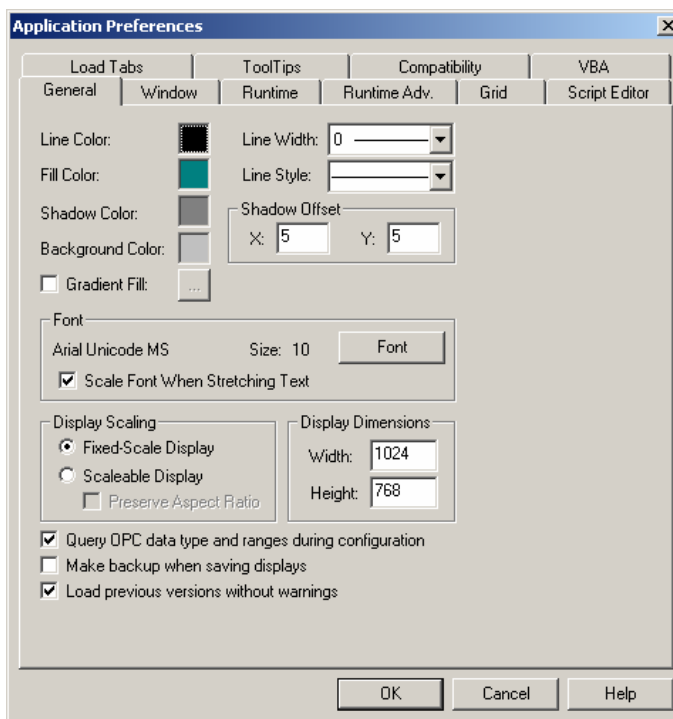


Figure 2.4. Application Preferences Dialog Box: General Tab

Besides **Line Color** and **Fill Color**, which allow you to choose the color you wish to use for defining lines and filling objects, **Background Color** provides choices of color for the background. **Shadow Color** and **Shadow Offset** change the depth of a selected object. You can also choose the **Line Width** and **Line Style**.

Click the **Font** button to select a font and point size for text. If you want to preserve the proportion of the text size relative to the display when resizing the display, check the **Scale Font When Stretching Text** check box.

Display Scaling allows you to choose between a **Fixed-Scale** display in which the objects will not change in size when the main window is resized, and a **Scalable Display** in which objects will automatically resize to fit within the main window. The **Preserve Aspect Ratio** check box provides you with the option to scale the display proportionate to the screen's aspect ratio.

The **Display Dimensions** settings allows you to decide the **Width** and **Height** of the working area of the display.

If the **Query OPC data type and ranges during configuration** check box (**Application Preferences** dialog box only) is checked, then the system will look for OPC data types and ranges during the configuration of the application.

To make a backup when saving your displays, check the **Make backup when saving displays** check box (**Application Preferences** dialog box only).

The **Save As Previous Version** feature allows you to save a display file created in a newer version of GraphWorX in one of the previous versions while preserving the existing version of the display file, so that the display file can be still loaded in the previous version of GraphWorX in which it was created. By default, GraphWorX provides warning messages about version compatibility. To suppress these version information messages when a display is loaded, check the **Load previous versions without warnings** check box (**Application Preferences** dialog box only). For more information, please see the Save As Previous Version feature.

Window Options

The **Window** tab in the **Application Preferences** and **Display Properties** dialog boxes, shown below, allows you to configure parameters for the GraphWorX main window.

Note: Some of the window settings are only active in runtime mode.

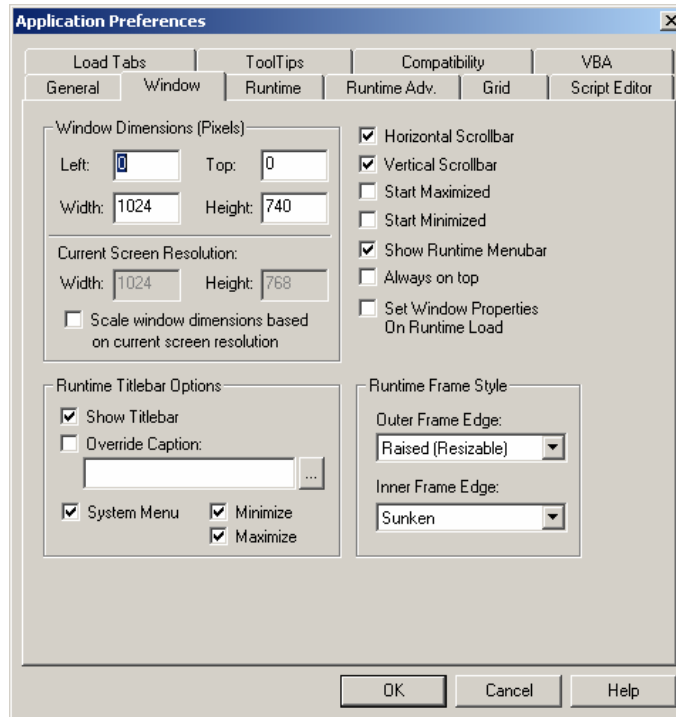


Figure 2.5. Application Preferences: Window Tab

The **Window Dimensions (Pixels)** field resizes the window to the exact dimensions desired (Left, Top, Width, and Height).

When the **Scale window dimensions based on current screen resolution** check box is checked, the window dimensions are automatically scaled based on the current screen resolution. For example, if the current screen resolution is 640x480 and your GraphWorX window size is 640x480, if you switch the window's resolution to 800x600 the next time your display is loaded, its dimensions will be scaled up to 800x600. If you want your window to always be 640x480 no matter what the resolution is, do not check this check box.

Note: If the screen resolution is changed, the window dimensions will remain the same as the way they were originally set up.

The **Runtime Title Bar Options** determines how the display window's title bar looks during runtime mode. You have the option to hide or show the title bar by unchecking or checking the **Show Title Bar** check box. To replace the title bar caption with a different caption, check the **Override Caption** check box and type a new caption in the text box below. To select an alias to use as a caption, click the ... button to the right of the text box and select either **Global Aliases** or **Language Aliases** from the pop-up menu, as shown in the figure below. This opens the respective alias browsers, allowing you to choose from a list of available aliases. You can also **Minimize** or **Maximize** the **System Menu** by checking the check boxes provided.

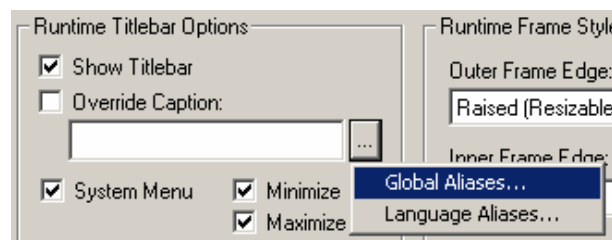


Figure 2.6. Selecting an Alias To Use for the Runtime Title Bar Caption

You may also choose to hide or show the **Horizontal/Vertical Scrollbars** (configuration mode and runtime mode), and the main **Menu Bar** (runtime mode only). You may choose to have the window start initially minimized or maximized. **Always on Top** ensures that the GraphWorX window will not be obscured by other open windows (runtime mode only).

Set Window Properties on Runtime Load indicates whether the window properties for a given display should be applied whenever that display is loaded while in runtime mode (when checked), or only when GraphWorX is initially launched into runtime mode (subsequent display loads will retain the window properties of the previously loaded display).

The **Runtime Frame Style** options allow you to specify styles for the inner and outer edges of the main window's border. In the list box under **Outer Frame Edge**, select None, Raised (Fixed Size), or Raised (Resizable). In the list box under **Inner Frame Edge**, select None or Sunken.

Runtime

In the **Runtime** tab in the **Application Preferences** and **Display Properties** dialog box, shown below, you can specify parameters for runtime mode:

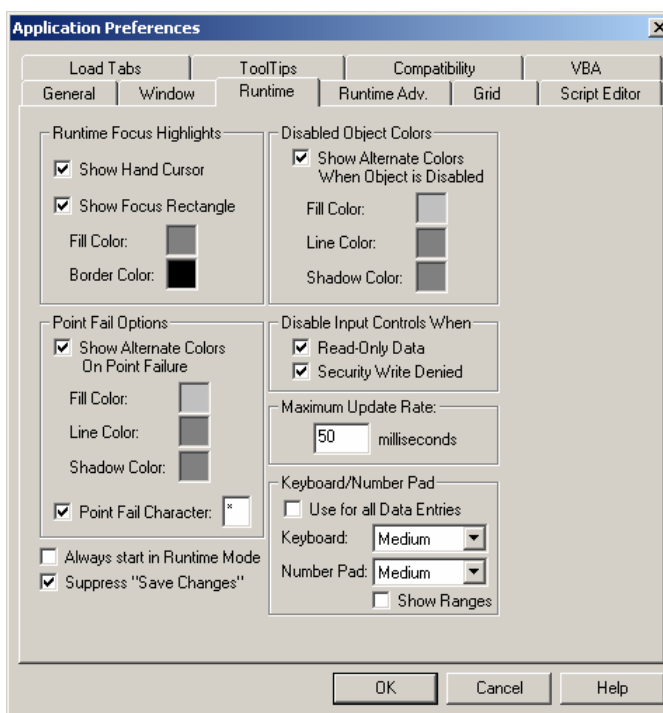


Figure 2.7. Application Preferences: Runtime Tab

For **Runtime Focus Highlights**, you can choose to show a **Hand Cursor**, which indicates that you can pick the object. You can also choose to show a frame or a **Focus Rectangle** around the pickable object, and specify a fill color and a border color for the frame.

Point Fail Options allow you to choose whether to **Show Alternate Colors** when data points fail, by checking the box provided. By default, the **Point Fail Character** (shown for process points and data entries) is an asterisk as shown above.

As with point fail options, **Disabled Object Colors** allow you to specify that you wish alternate colors to be shown when the object is disabled. Like disabled buttons in Windows, when an object is disabled in GraphWorX, you cannot pick that object. **Data Input Controls** can be automatically disabled when the associated data connection is **Read-Only** or when **Security Write Access** is denied for the associated data connection.

The **Maximum Update Rate** reflects how fast runtime updates can occur. The fastest possible update rate is 50 milliseconds.

The **Always Start in Runtime Mode** check box (**Application Preferences** dialog box only) allows you to choose to start the application in runtime mode. Checking the **Suppress "Save Changes"** check box (**Application Preferences** dialog box only) saves the changes made in runtime without prompting the user to do so.

Keyboard and Number Pad

The **Keyboard/Number Pad** field (**Application Preferences** dialog box only) enables you to globally configure ranges for keyboards and number pads, such as the one shown below, for use with entering values to process points during runtime mode. If the entry is text, the keypad opens; if the entry is a number, the number pad opens. The available ranges in the **Keyboard** and **Number Pad** list boxes are Very Small, Small, Medium, Large, and Very Large. Checking or unchecking the **Show Ranges** check box will show or hide ranges in the number pad dialog box. The number pad can be optionally shown in the **Data Entry-Keypad** function of the **Process Point/Data Entry (PPT/DE)** tab of the **Property Inspector** dialog box.

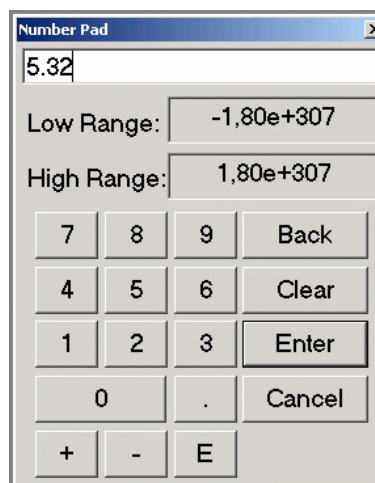


Figure 2.8. Number Pad

Resizable Font

The fonts inside the keypad and the number pad are now resizable. The font size is increased when size of the keypad increases and the font size is decreased when the size of the keypad decreases.

Caps Lock Switch and Support for International Languages

Some keys have been aggregated together on one unique key. When you push the **Caps Lock** key, the keypad layout changes and shows the new character in CAPS and the new special character (for example, the key <<a>> becomes <<A>> and the key <<2>> becomes <<@>>), as shown in the figure below. New keys have been added for German, Italian and French and other special characters.

The **International** key on the keypad acts like a check box (like the **Caps Lock** key); if you click it then it remains depressed until the next time you click it. When the **International** key is not pressed (default) the keypad hides all the buttons containing characters that do not appear on the default English keyboard layout. When the **International** key is pressed, then the keypad does not hide any keys and the full layout is displayed with, for example, the German, Italian, and French special characters, as shown in the figure below.

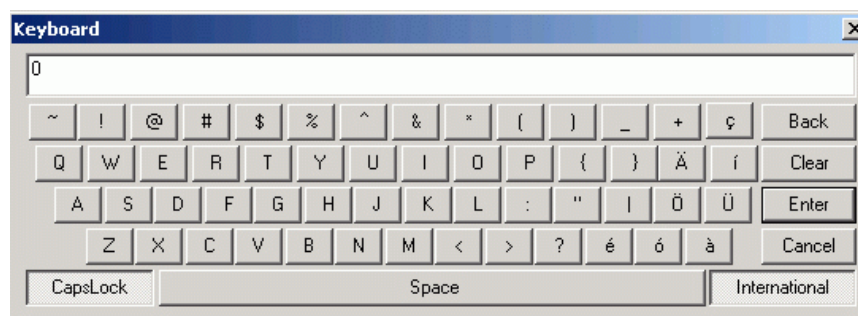


Figure 2.9. Keypad with Uppercase Characters

The language switching for the number pad has been fixed so that when you switch languages the separator is a decimal separator and the thousand separator changes. For example:

- US notation: 15,030.99
- European notation: 15.030,99

This is reflected in the number pad key, which is <<.,>> (point) or <<,>> (comma) in function of the language currently selected in GraphWorX.

Customizing the Keypad

As usual it is possible to customize the keypad through the Microsoft Visual Studio resource editor. You can specify up to two different characters for each key. Clicking the Caps Lock button switches between the two sets of characters. The first character is used when Caps Lock is clicked. The second character is used when Caps Lock is not clicked. The two characters do not need to be correlated: for example, <<2>> and <<@>>. If you do not specify a second character for a key, then the corresponding lower case system key will be used.

Runtime Window Properties Mode

Selecting **Runtime Window Properties Mode** from the **View** menu makes it possible to see what a display will look like in runtime mode without actually entering runtime. This helps you set the proper display dimensions and zoom factors. When the action toolbars are hidden, the configuration functions can be found in the **Main** menu or in the right-click popup menu.

Note: Pressing **CTRL+M** will toggle the display between configuration and runtime modes.

Runtime Advanced Settings

Application Preferences

The **Runtime Advanced** tab in the **Application Preferences** dialog box, shown below, specifies the following application parameters for runtime mode.

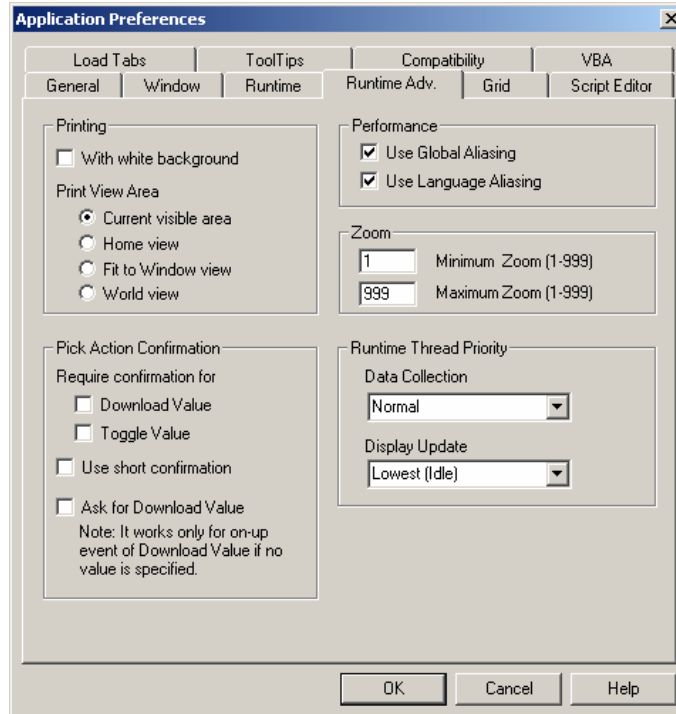


Figure 2.10. Application Preferences: Runtime Advanced Tab

Printing and Optional Print Area

The **Printing** field contains options for printing out a GraphWorX display. Checking the **With white background** check box overrides the default background color so that the display is printed with a white background.

The **Print View Area** options determine how much of the display will appear on the printout:

- Current visible area
- Home view
- Fit to window view
- World view

Pick Action Confirmation Messages

The **Pick Action Confirmation** field provides an option to require confirmation messages for the **Download Value** and **Toggle Value** pick actions. You can minimize the message length by checking the **Use Short Confirmation** check box.

Note: If no value is specified for the On-Up event, the Download Value pick action will display a number pad or a keypad dialog in runtime, where you can enter the value. This feature must be manually enabled in the **Runtime Advanced** tab of the **Application Preferences** dialog box by checking the **Ask for Download Value** check box.

Performance

The **Use Global Aliasing** and **Use Language Aliasing** check boxes under the **Performance** section are used to enable or disable support for global aliasing and language aliasing.

Zoom Preferences

The **Zoom** section allows you to specify the **Minimum Zoom** and the **Maximum Zoom** levels. It is possible to input any number between 1 and 999 for both the minimum and maximum zooms. If you enter a minimum zoom larger than the maximum zoom, GraphWorX automatically swaps the two numbers so that the maximum zoom is always greater than the minimum zoom.

Runtime Thread Priority

The **Runtime Thread Priority** sets the priority levels for data collection and display updates during runtime mode. Select one of the following priority levels from each drop-down list under **Data Collection** and **Display Update**:

- Lowest (Idle)
- Low
- Below Normal
- Normal
- Above Normal
- High
- Highest (Time Critical)

Display Properties

The **Runtime Advanced** tab in the **Display Properties** dialog box, shown below, specifies the following display parameters for runtime mode.

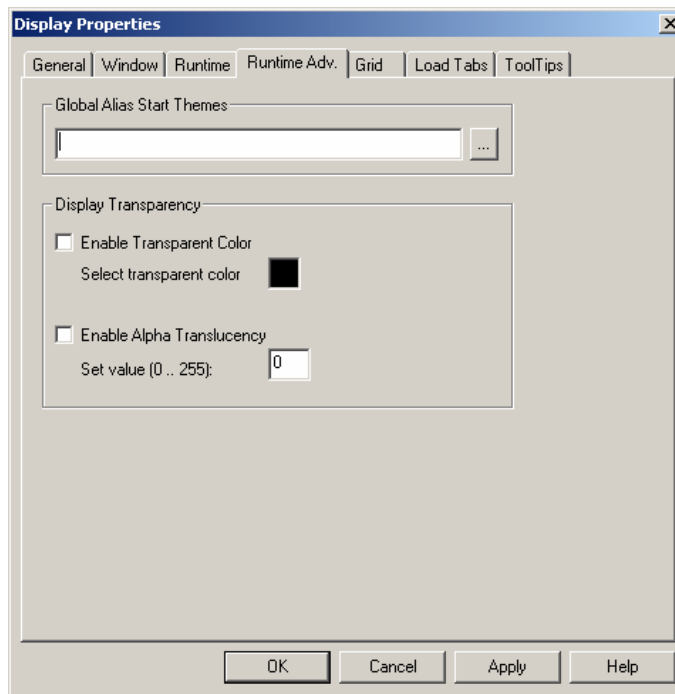


Figure 2.11. Display Properties: Runtime Advanced Tab

Global Aliasing Theme Scope

The resolution of aliases is controlled by the theme scope (i.e. on what level a theme is applied to your application). The Global Aliasing system supports three different levels of aliases, as shown in the figure below:

- Machine-Wide: Applies to the whole computer.
- Process-Wide: Applies to the current process (e.g. GraphWorX with pop-up windows).
- Document-Wide: Applies only to the current document/display and its embedded controls (e.g. a GraphWorX pop-up window).

The aliases are assigned to a certain group based on the following delimiters:

- <!MACHINE!>
- <@PROCESS@>
- <#DOCUMENT#>

In GraphWorX, you can have multiple levels of documents, or pop-up windows, as shown in the figure below.

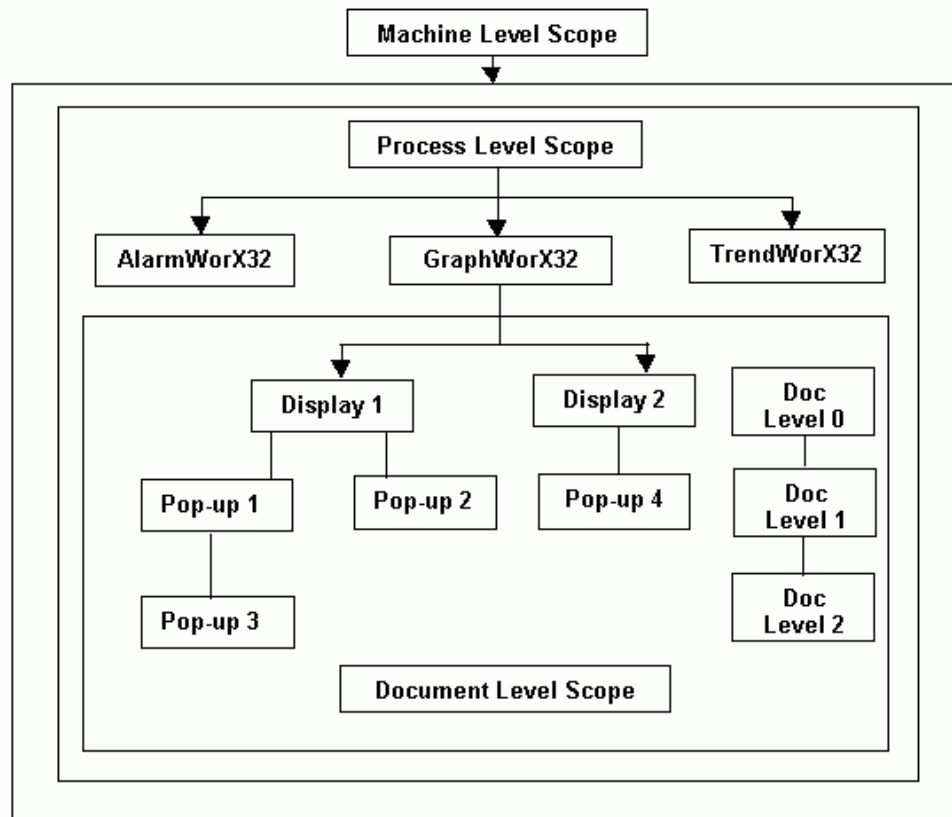


Figure 2.12. Overview of Themes Scopes

The document level theme scope is defined in the GraphWorX **Display Properties**. To define the default theme scope in GraphWorX:

1. Either type a theme name in the text field under Global Alias Start Themes or click the ... button to select a theme from the Themes dialog box, as shown in the figure below.

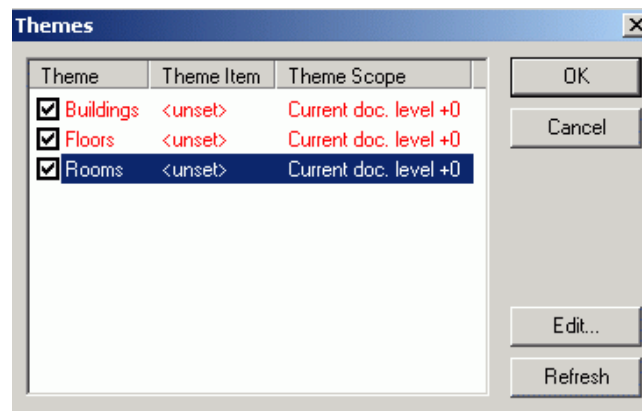


Figure 2.13. Choosing a Global Alias Start Theme

2. To define the theme scope, highlight a theme and click the **Edit** button. This opens the **Theme Editor** dialog box, as shown in the figure below. Select a **Theme Item** (e.g. "Room1") to associate with the start theme. Under **Theme Scope**, you can specify an **Absolute** theme scope (e.g. machine level, process level, or document level). Alternatively, you can specify a scope for the theme that is **relative to the current document level** (e.g. main display, pop-up window 1, pop-up window 2, etc.). Click **OK**.

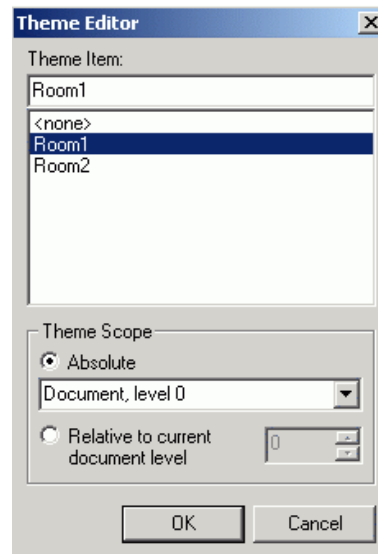


Figure 2.14. Specifying the Theme Item and Scope

3. The selected **Theme Item** and **Theme Scope** are now indicated in the **Themes** dialog, as shown in the figure below. Click **OK**.

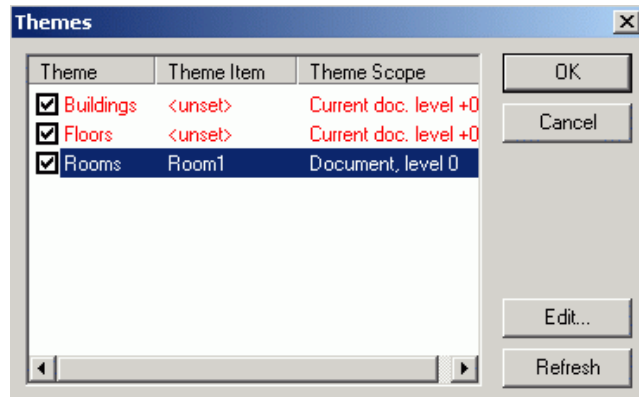


Figure 2.15. Theme Scope Indicated

4. The selected themes appear under Global Alias Start Themes in the Display Properties dialog box, as shown in the figure below.

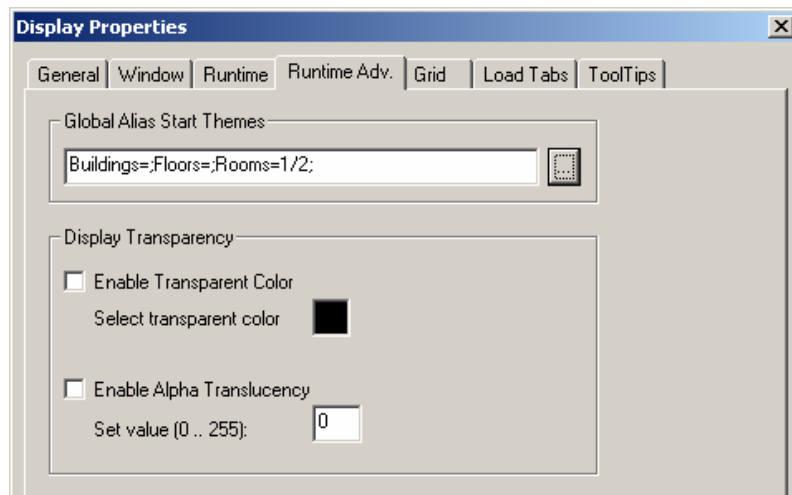


Figure 2.16. Global Alias Themes Added to Display Properties

Transparency and Translucency

A new feature of GraphWorX (supported on Windows 2000 and later) enables defined runtime display transparency and translucency.

To enable display transparency, check the **Enable Transparent Color** check box. Choose the color that will be transparent during runtime mode (black by default) using the **Select transparent color** box.

To enable display translucency (also called alpha blending), check the **Enable Alpha Translucency** check box. In the **Set value** field, the translucency can be set in a range 0-255. When the alpha value is 0, the window is completely transparent. When the alpha is 255, the window is opaque.

In addition to setting the values in the **Display Properties**, the following automation methods, members of **Gwx.GwxDisplay**, set the transparency in runtime:

- Function SetTransparency(EnableTransparency As Boolean, transparentColor As OLE_COLOR, EnableAlpha As Boolean, AlphaBlending As Integer) As Boolean
- Function GetTransparency(TransparencyEnabled As Boolean, transparentColor As <Unsupported variant type>, AlphaEnabled As Boolean, AlphaBlending As Integer) As Boolean

Grid

The **Grid** tab in the **Application Preferences** and **Display Properties** dialog boxes, shown below, gives you flexibility with line style, color, and spacing. The **Grid Size** is in arbitrary base units (depending upon the current zoom factor). When zoom is at 100%, the spacing is in pixels.

On the **Grid** tab, **Grid Style** allows you to choose a style of dots or dashes (lines). You can choose a **Vertical Line Style** or a **Horizontal Line Style** from the options available. The **Grid Properties** specify the grid spacing in terms of width and height.

You can choose a **Grid Color** for the lines and dots of the grid. By checking the appropriate boxes, you can choose to **Show Grid** and **Snap to Grid**. The **Grid** command on the **View** menu also shows or hides the grid.

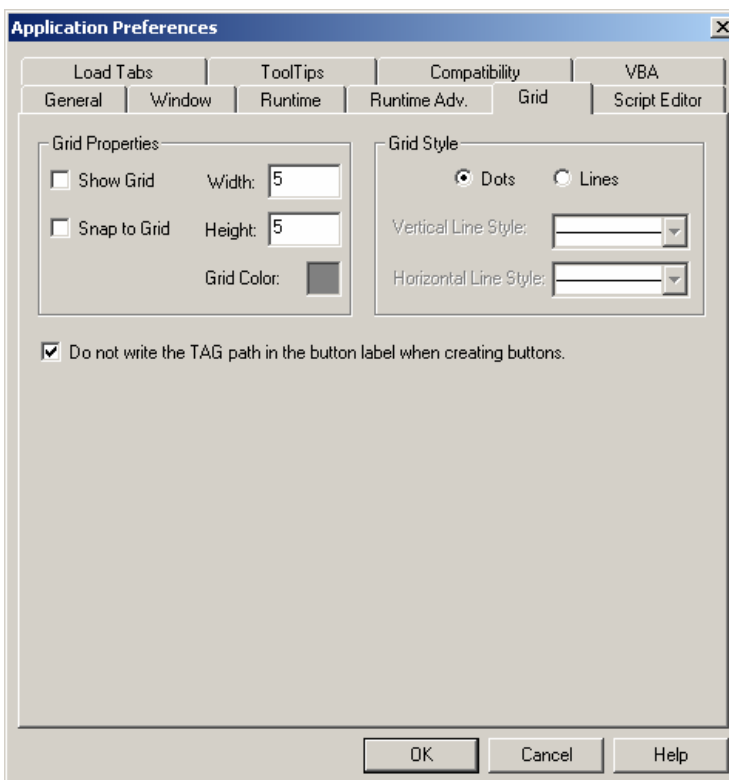


Figure 2.17. Application Preferences: Grid Tab

On the **Grid** tab, **Grid Style** allows you to choose a style of dots or dashes. **Grid Size** allows you to specify the grid spacing in terms of width and height. You can choose a **Vertical Line Style** or a **Horizontal Line Style** from the options available.

You can choose a color for the lines and dots of the grid. By checking the appropriate boxes, you can choose to **Show Grid** and **Snap to Grid**.

Script Editor

The Script Editor is designed to be customizable for users with special needs. The customization settings can be configured in the **Script Editor** tab of the **Application Preferences** dialog box. Select **Application Preferences** from the **Format** menu in GraphWorX. This opens the **Application Preferences** dialog box, as shown in the figure below. Click on the **Script Editor** tab. You can change the following Script Editor settings, as shown in the figure below:

- Font face, size and style
- VBScript keywords and constants
- JScript keywords and constants
- Color for keyword, constants, string, number, and comment
- Enable or disable the syntax coloring
- Enable or disable the automatic case change feature

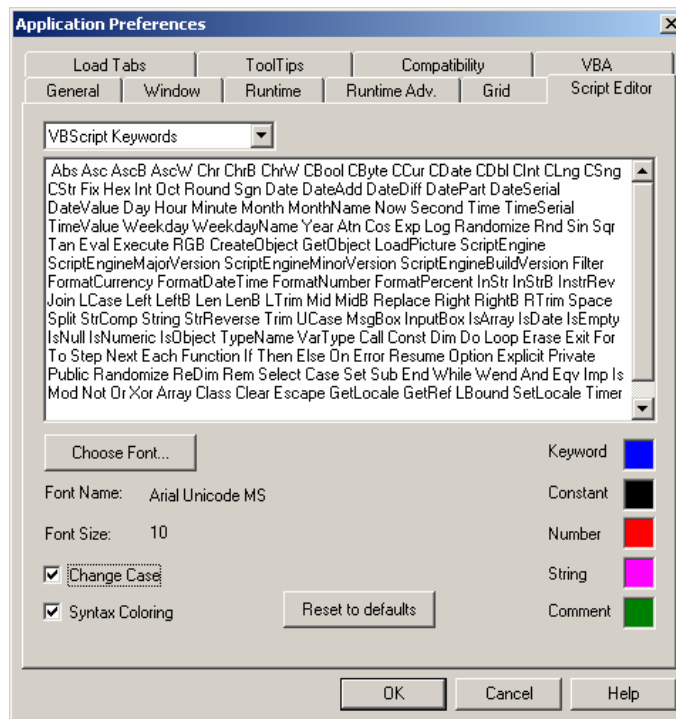


Figure 2.18. Application Preferences: Script Editor Tab

Load Tabs

The **Load Tabs** tab in the **Application Preferences** and **Display Properties** dialog boxes, shown below, allows you to construct a row of tabs; each tab will load a display file. Load display tabs are a convenient method for navigating through numerous display files.

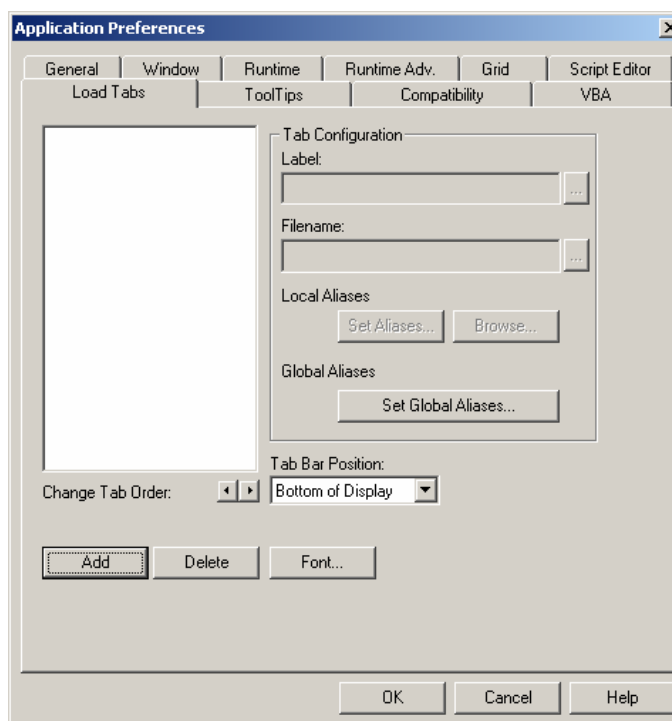


Figure 2.19. Application Preferences: Load Tabs Tab

To add a tab, click **Add** and select one or more files from the **Assign File to New Tab** dialog box, as shown in the figure below. Click **Open**. Tabs are added in the list box on the left-hand side of the **Load Tabs** properties page, as shown above. To delete a tab from the list, click the **Delete** button. To change the position of the tabs in the list (move them up or down), click the arrow buttons next to **Change Tab Order**. You can also change an existing tab's settings by double-clicking on it inside the list box or by selecting the tab in the list and clicking the **Browse** button.

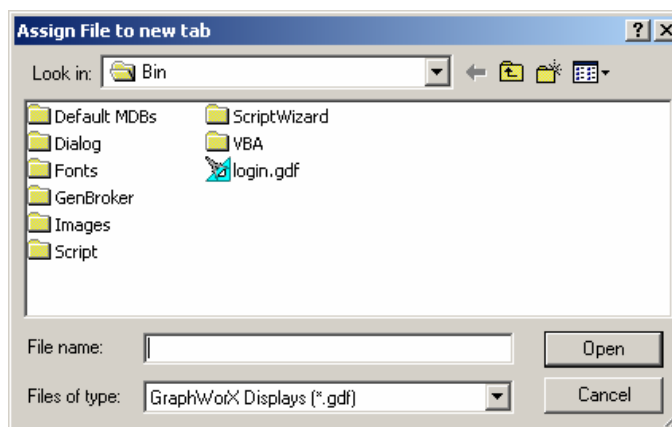


Figure 2.20. Assigning a File to a Tab

Select a **Tab Bar Position** at the top or at the bottom of the display. In the **Tab Configuration** section, type a **Label name** that for each tab that will identify the tab in the display. You can choose a **Font** from the options available by clicking on the **Font** button. The file name and directory path for each tag are listed under **File name**.

You can also specify both local and global aliases. Under **Local Aliases**, click the **Set Aliases** button. This opens the **Set Aliases Configuration** dialog box, where you can define the alias parameters. Under **Global Aliases**, click the **Set Global Aliases** button to select a theme from the **Themes** dialog box, as shown in the figure below. Click **OK**.

Note: To edit a theme's properties, select a theme and click the **Edit** button. This opens the **Theme Editor**, where you can specify theme items and the theme scope. For information about editing global alias themes and specifying the theme scope, please see the **Runtime Advanced Settings** section above.

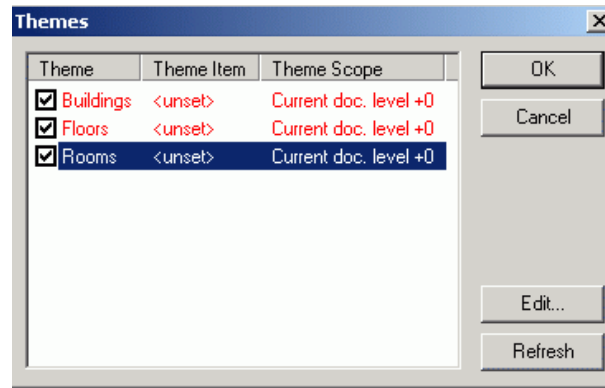


Figure 2.21. Choosing Global Alias Themes

ToolTips

ToolTips are text messages that pop up when the mouse is moved over certain objects in a display. The **ToolTips** tab in the **Application Preferences** and **Display Properties** dialog boxes is shown below.

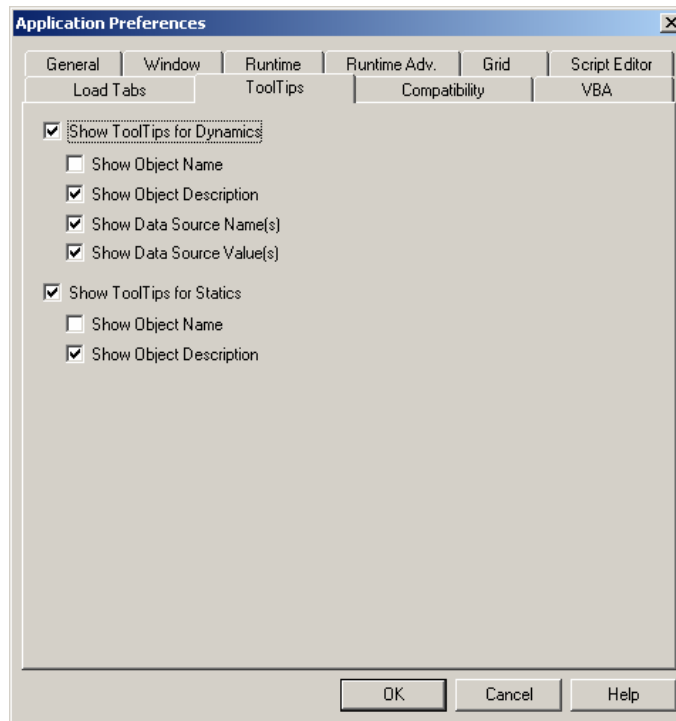


Figure 2.22. Application Preferences: ToolTips Tab

ToolTips are enabled by default, but you can customize the ToolTip settings. The **Show ToolTips for Dynamics** section provides the following ToolTip options that apply to dynamic connections in the display (such as size dynamics or color dynamics):

- Show Object Name
- Show Object Description
- Show Data Source Name(s)
- Show Data Source Value(s)

The **Show ToolTips for Statics** section provides the following ToolTip options that apply to static objects in the display (such as rectangles and ellipses):

- Show Object Name
- Show Object Description

Compatibility

The **Event Log** enhancement adds more detailed event log reporting to GraphWorX. In addition to existing log information, GraphWorX now also logs the following actions:

- When GraphWorX starts
- When GraphWorX is shut down
- When a user saves a GraphWorX file

When a user saves a GraphWorX file, the message contains the full path of the file changed and the name of the user(s) that are logged into the Security Server at the time.

GraphWorX now properly fills-in the **variant prevVal** in the Event Log structure when logging a writing to OPC data from a GraphWorX display, and it allows you to specify the "Source" name, which in previous version of GraphWorX was always the name of the application. To configure the source type:

5. Select Application Preferences from the Format menu in GraphWorX. This opens the Application Preferences dialog box, as shown in the figure below. Click on the Compatibility tab.

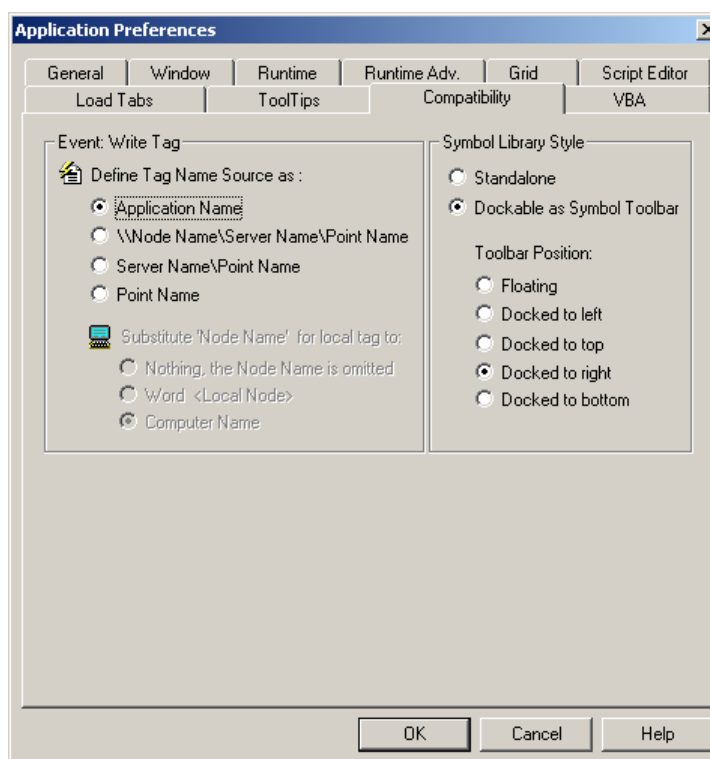


Figure 2.23. Application Preferences: Compatibility Tab

6. Select a source name, which can be one of the following values:
 - Application name (just like it worked in earlier versions)
 - \\NodeName\\ServerName\\PointName (When this option is selected, you have the option to omit the "NodeName" from the string to substitute the "NodeName" with the local computer's name, as shown in the figure below.)
 - ServerName\\PointName
 - PointName
7. Click OK. The current configuration is stored to the registry.

Visual Basic for Applications

Runtime Mouse and Keyboard Events

Early versions of GraphWorX (prior to version 5.0) contained mouse and keyboard events, which were fired to Visual Basic for Applications (VBA) and to the host container. Due to performance reasons, these events were removed in subsequent versions.

GraphWorX version 7 enables them again, but to avoid performance issues, events can be enabled individually, either globally or per display. The mouse and keyboard runtime events can be selected globally for all instances of GraphWorX on given computer:

8. Select Application Preferences from the Format menu in GraphWorX. This opens the Application Preferences dialog box, as shown in the figure below. Click on the VBA tab.
9. Select the desired mouse/keyboard events, which are then fired by GWX/GWXview to VBA or to the host container. Click OK. The settings are saved to the registry when GraphWorX stops and are restored when GraphWorX starts.

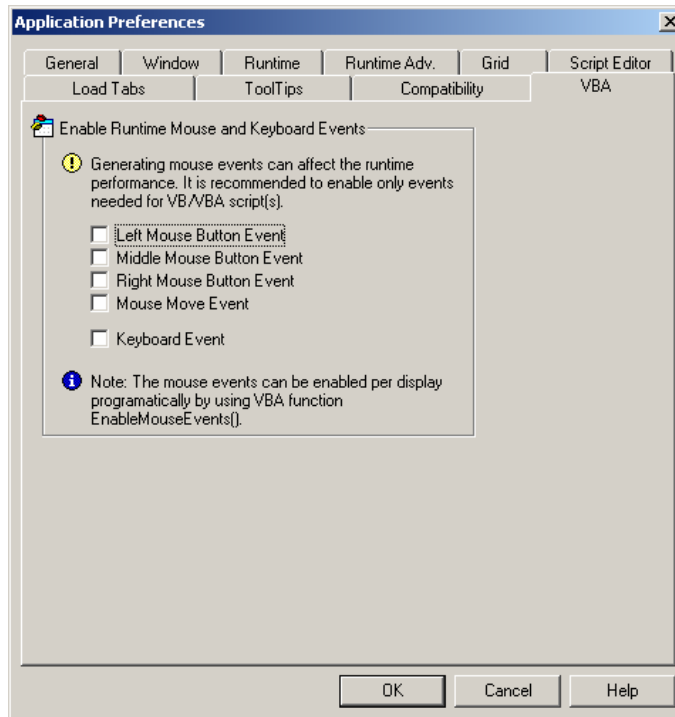


Figure 2.24. Application Preferences: VBA Tab

Configuring Runtime Events Locally (Per Display)

The mouse and keyboard runtime events can be also selected locally for a particular display(s) by programmatically using the following automation function:

```
Sub EnableRuntimeEvents( _  
    LeftButton As Boolean, MiddleButton As Boolean, RightButton As Boolean, _  
    MouseMove as Boolean, Keyboard as Boolean)
```

A typical use of this method:

```
Private Sub GwxDisplay_PreRuntimeStart()  
    ' Enable right click events only  
    Call ThisDisplay.EnableRuntimeEvents(False, False, True,  
    False, False)  
End Sub
```

Copying and Resetting Preferences and Properties

You can reset application preferences and display properties to their default settings by choosing the corresponding **Reset Default Application Preferences** or **Reset Default Display Properties** from the **Format** menu.

You can also copy settings from preferences to properties, or from properties to preferences. Choosing **Save Properties As Preferences** from the **Format** menu will open the **Choose Settings** dialog box, shown below:

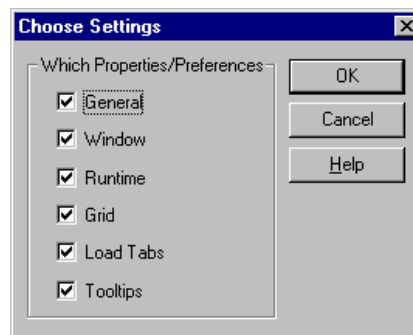


Figure 2.25. Choose Settings Dialog

Choosing **Apply Preferences to Properties** from the **Format** menu also opens the **Choose Settings** dialog box.

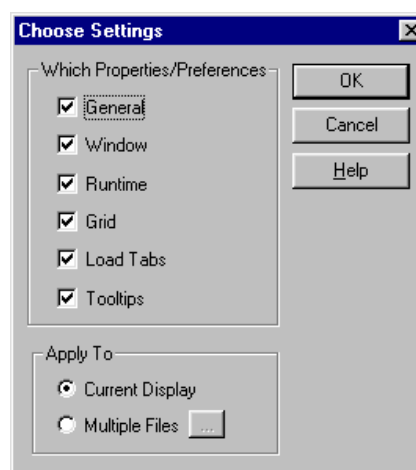


Figure 2.26. Choose Settings Dialog

Each check box in the dialog corresponds to a properties/preferences category. Only those categories that are checked will be copied when **OK** is clicked. To apply the application preferences to the current display only, select **Current Display**. To apply the application preferences to more than one display, select **Multiple Files** and then click the ... button. The Select Files dialog box opens, as shown in the figure below. Click the **Add** button to select files to which to apply application preferences. The files appear under the **File List** field.

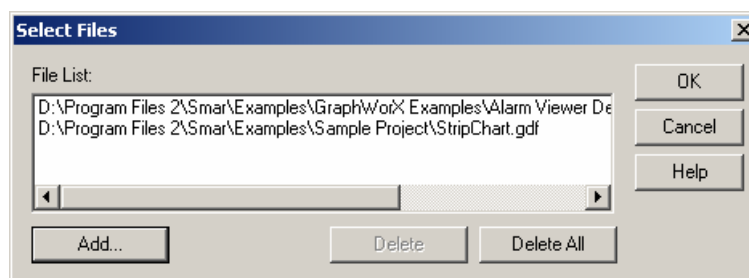


Figure 2.27. Selecting Files for Applying Preferences

Language-Aliasing Support

GraphWorX has all of its strings saved in the resource file, which can be modified to provide custom versions. In addition, it provides real-time support for loading a resource-only .dll to support an international language. This .dll file is compiled as a standard Win32 .dll. Any time the Smar applications settings are switched to a different language, GraphWorX will load the corresponding resource .dll.

The Unicode version of GraphWorX also supports language aliasing with automatic range and value scaling, as well as unit conversions, formatting, and font selection. For more information about configuring Unicode version language-aliasing support, refer to the **ProcessView Language Configurator** documentation.

Note: To enhance language-aliasing performance, select the Microsoft Arial Unicode font, which contains all Unicode characters. When you begin configuring in GraphWorX, you should first select **Application Preferences** from the **Format** menu in GraphWorX. Click the **Font** button in the **Design** tab of the **Application Preferences** dialog box and select the Arial Unicode MS font. The Arial Unicode MS font must be selected separately within the Properties dialog boxes of each ActiveX component.

Sample Language Configuration

The Unicode Installation of ProcessView installs a sample Language Server configuration under the "Languages" folder of the ProcessView product installation "tree." In addition, a sample display "languagesDemo.gdf," which includes a sample Viewer configuration, is provided under the "Languages" folder. The figure below shows this sample language-aliasing demo in runtime mode.

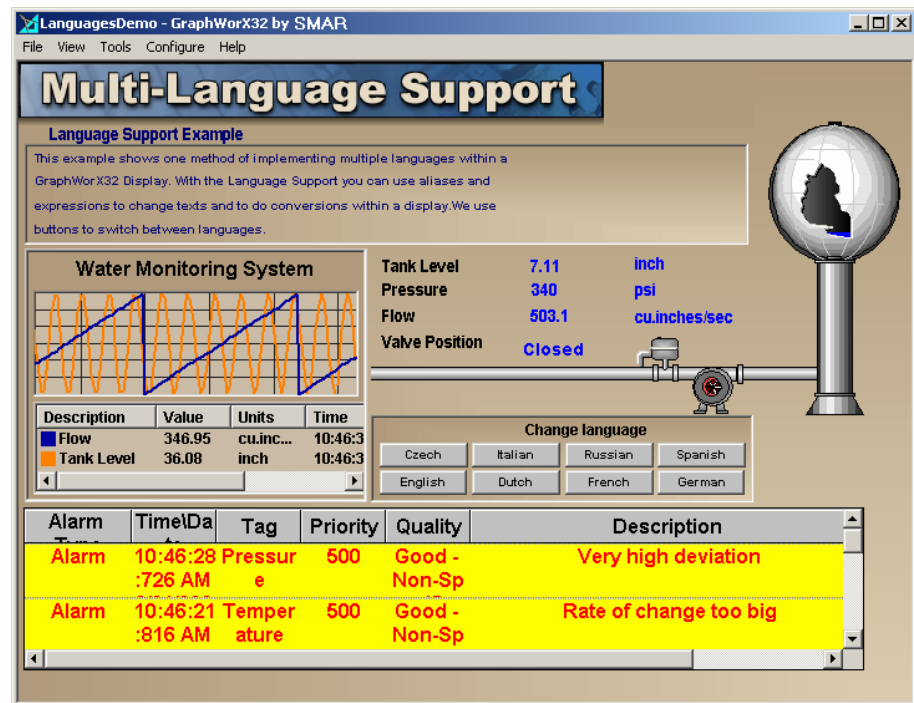


Figure 2.28. Language-Aliasing Demo in Runtime Mode

This figure below shows this same language-aliasing demo in configuration mode.

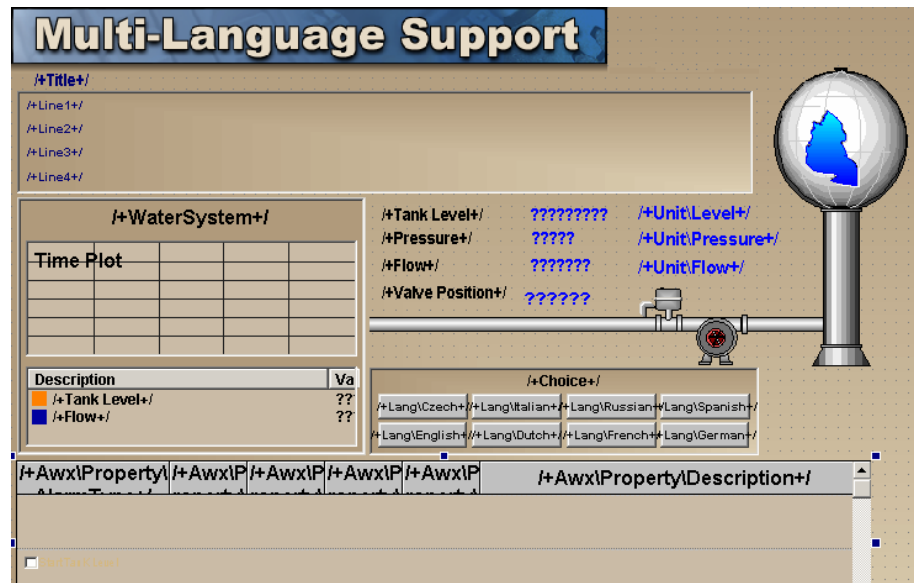


Figure 2.29. Language-Aliasing Demo in Configuration Mode

For the example above, when you click on the "Tank Level" text box, the **Property Inspector** dialog box opens, as shown in the figure below.

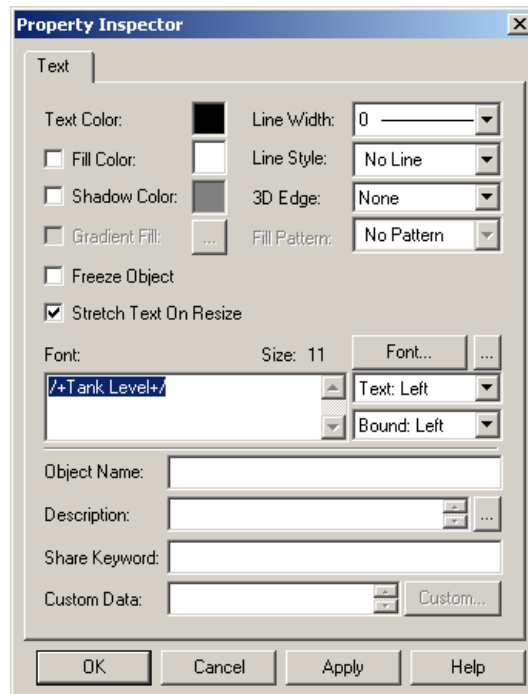


Figure 2.30. Property Inspector: Text

Similarly, when you click on the process point for "Tank Level," the **Property Inspector** dialog box opens as shown below. The language-aliased string "Unit\Level" appears at the bottom of the **PPT/DE** tab of the **Property Inspector** dialog box in the **Lang Alias** field, as shown in the figure below. Note that the strings within a "/" and "-/" delimiter pair define a language-aliased string.

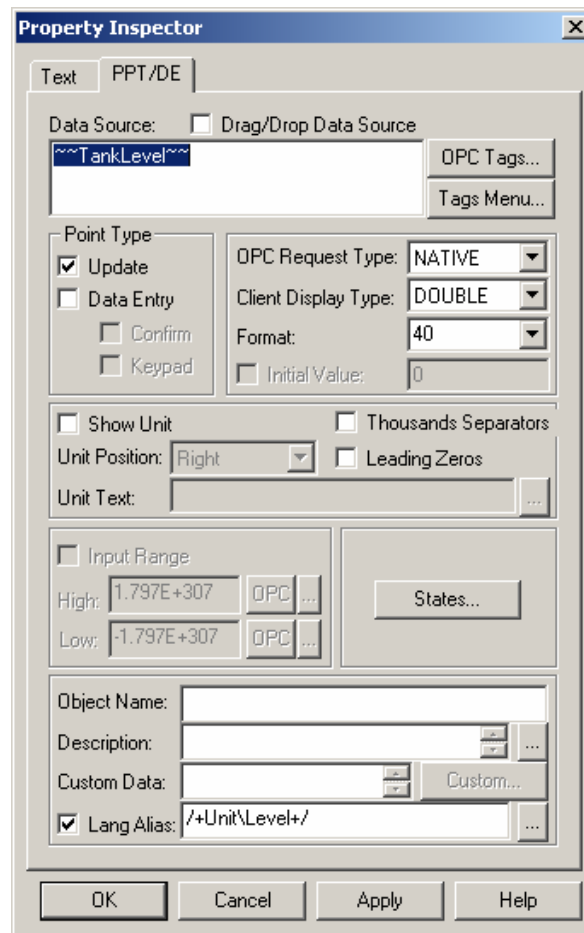


Figure 2.31. Property Inspector: Process Point

Note: The Unicode version language aliasing is independent of the resource .dll selection, which relates mostly to the text in the dialog boxes and other user interface elements.

Creating Language-Aliased Strings

Aliases can also be used in the Property Inspector to provide the related aliases for unit scaling, font selection, and date/time translation. During runtime mode, GraphWorX interfaces to the Smart Language Server, and it tries to resolve the language-aliased strings. The following examples demonstrate how to configure language-aliased strings in GraphWorX.

Creating a Language-Aliased Text String

In general, you can use language aliases in GraphWorX almost everywhere text is typed. To create a language-aliased text string, do the following:

1. In configuration mode in GraphWorX, select **Text** button on the **Draw** toolbar to create a text box. A text box will appear in the display.
2. Type the desired text in the text box. The text should be preceded by the "/" delimiter and followed by the "/" delimiter. For example, the word "hello" would be typed as "/+hello+/" in the text box, as shown in the figure below. The delimiters tell the Language Server that the word "hello" is a language-aliased text string.

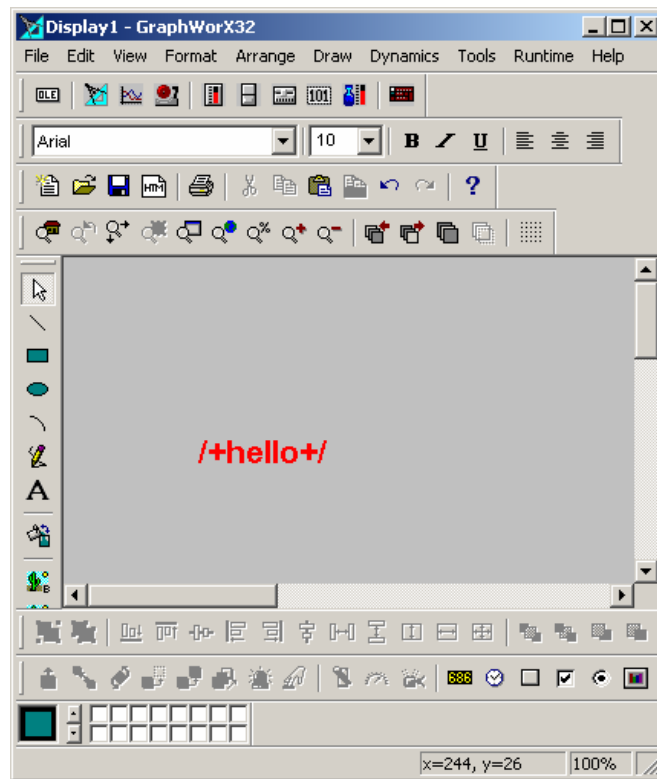


Figure 2.32. Text With Delimiters

3. Double-click on the text box to open the Property Inspector dialog box, as shown below. Notice that "/+hello+/" shows up in the text field of the Property Inspector.

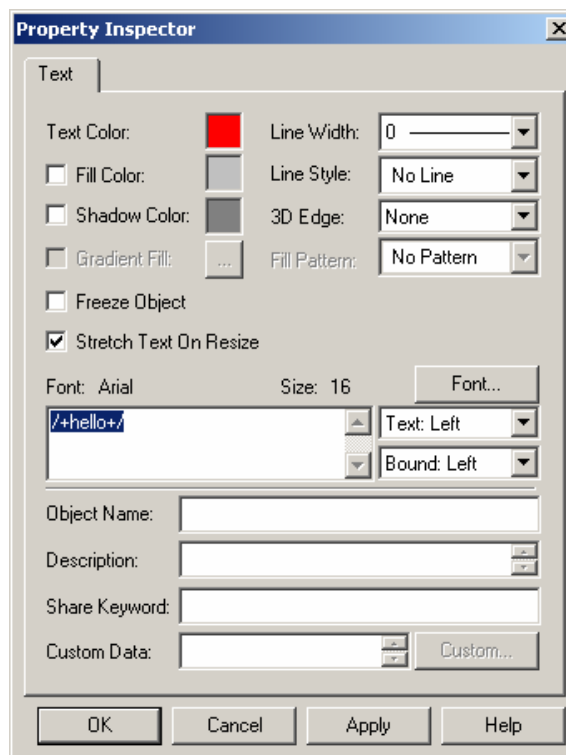


Figure 2.33. Property Inspector With Language-Aliased Text String

4. Open the Language Configurator from the Windows **Start** menu by selecting **Programs > System302 > ProcessView > Languages > Language Server Configurator**. Configure a new language alias in the language database for the text string you just created. If you do not create an alias for the text, the Language Server will not know how to translate the text. When

you create the alias in the Language Configurator, the alias name in the Language Configurator must exactly match that of the text in the GraphWorX text box. So in this case, the alias name will be "hello." You can add as many languages as you like to the alias. For each language you add to the alias for "hello," you can enter the translation text for that language. For example, you might use "Hello, how are you?" as the English translation text for the "hello" alias, and you might use "Guten tag" as the German translation for the "hello" alias. Refer to the **Language Configurator** help documentation for detailed information about configuring language aliases.

5. When you have finished configuring the translation for the language alias, enter runtime mode in GraphWorX. The translated text for "hello" ("Hello, how are you?") now shows up in the GraphWorX runtime display, as shown below. Any other language translations that you configure for the alias will also be available in runtime mode. To switch between languages, choose **Select Language** from the **View** menu of GraphWorX in runtime mode. This opens the **Language Selector** dialog box. Choose the language you would like to have displayed. In this case, if you choose German, "Guten tag" will show up in the display.



Figure 2.34. Language Translation in Runtime Mode

Creating Other Language-Aliased Strings

Language aliases can also be used in the Property Inspector to provide the related aliases for unit scaling, font selection, and date/time translation. For example, let's assume that you have configured a language-aliased text string in GraphWorX for `"/+inches+/"` using the method described above. Using the Smar Language Configurator, you can create an "inches" alias for the text string. Let's suppose that, as in the example above, you want to enter English and German translations for the "inches" alias. For example, you might use "inches" as the English translation text for the "inches" alias, and you might use "cm" (centimeter) as the German translation for the "text" alias.

1. If you want to associate the "inches" alias with a dynamic action in GraphWorX, such as a process point, you need to tell the Language Server how to convert between inches and centimeters. Thus, you must also associate the alias with an expression that defines the conversion. First you need to create a new expression using the Smar Language Configurator. Then you need to add this expression to the "inches" alias. For more details about how to add an expression, refer to the **Language Configurator** help documentation.
2. Once you have added the unit conversion expression to the "inches" alias, you can now associate the alias with a process point. In GraphWorX, select **Intrinsics > Process Point** from the **Dynamics** menu. This adds a process point to the display.
3. Click on the GraphWorX display to open the Property Inspector dialog box. Click on the **PPT/DE** tab of the Property Inspector. Select a data source. In this example, we will use the "gfwsim.ramp.long" simulation variable, as shown in the figure below.

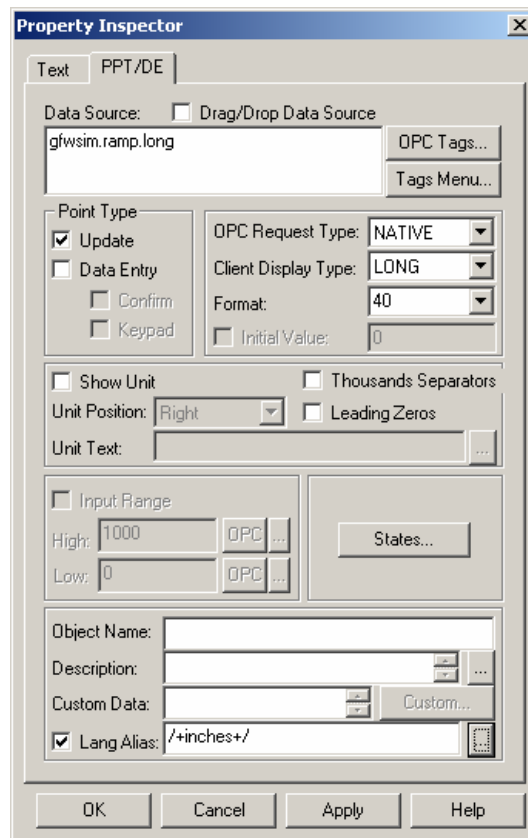


Figure 2.35. Process Point With Language-Aliased String

4. Check the **Lang. Alias** box at the bottom of the Property Inspector, as shown below. This enables the Language Alias field.
5. Enter the "/+inches+/" language alias string in the **Language Alias** field, as shown below. Click **Apply**. Now the process point is associated with the "inches" alias.

Note: You can also select a language alias from the Language Alias Browser by clicking the ... button next to the **Lang Alias** box and selecting Language Aliases. For more information about using the Language Alias Browser, please see the **Selecting Language Aliases From the Language Alias Browser** section.

6. When runtime mode is activated, you will see the dynamic process-point values for "gfwsim.ramp.long." To switch between languages, choose **Select Language** from the **View** menu of GraphWorX in runtime mode. This opens the **Language Selector** dialog box. Choose the language you would like to have displayed. If English is selected, the process-point values will be in *inches*. If German is selected, the values will be in *centimeters (cm)*.

Selecting Language Aliases From the Language Alias Browser

When specifying a language alias in the Property Inspector, you can also select a language alias from the Language Alias Browser, which includes all language aliases in the language database. This eliminates the need to manually type in the alias name.

1. In the Property Inspector (e.g. for a process point), check the **Lang Alias** check box and click the ... button next to the **Lang Alias** box and select **Language Aliases**, as shown in the figure below.

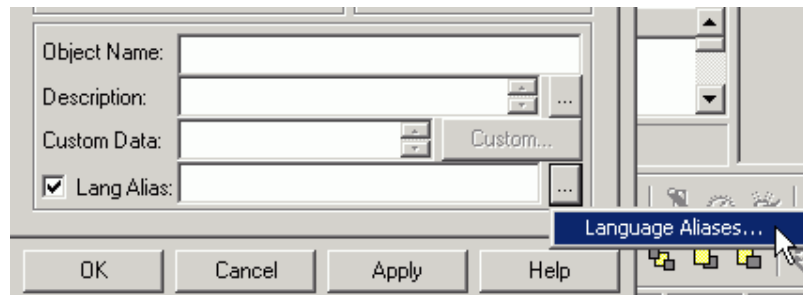


Figure 2.36. Opening the Language Alias Browser

2. The **Language Alias Browser** opens, as shown in the figure below. The browser includes all languages aliases in the language database. All language aliases that are configured in the Language Configurator are conveniently available to choose from inside the browser. The tree control of the Language Configurator is mimicked in the tree control of the Language Alias Browser. Select a language alias by double-clicking the alias name (e.g. "Line 2" in the figure below). The alias name appears at the top of the browser, which automatically adds the /+ and +/- delimiters to the alias name. Click the **OK** button.

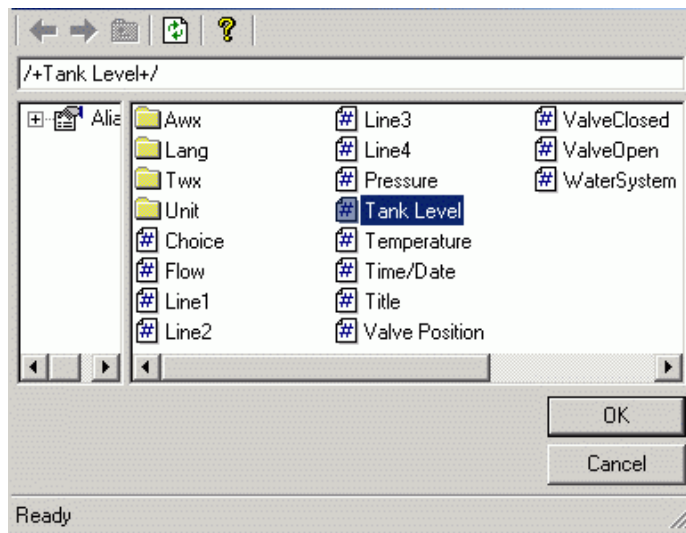


Figure 2.37. Selecting an Alias From the Language Alias Browser

3. The language alias (with delimiters) is added to the **Lang Alias** field of the Property Inspector dialog box, as shown in the figure below.

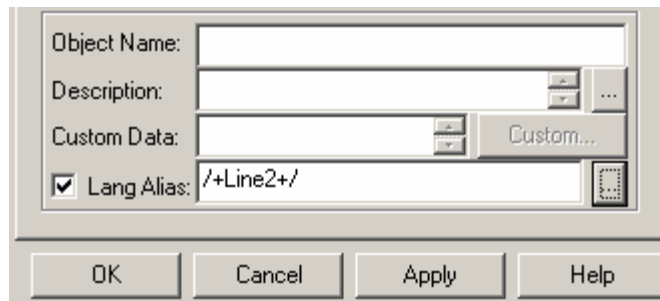


Figure 2.38. Language Alias Added to Property Inspector

Global Aliasing Support

GraphWorX supports document level, process level, and machine level global global aliasing. A **global alias** is a data string that enables you to reference multiple data sources (e.g. a process point in GraphWorX) with one unique name. Because you can reference multiple data sources from a single location, global aliasing can reduce the overall number of individual display files you need to create. Global aliasing is integrated into the following GraphWorX features:

- Support for visible objects (rectangles, ellipses, lines, etc.)
 - Description
 - Custom command
 - Global Alias Browser for general page
- Support for dynamics (size, location, rotation, etc.)
 - OPC Tag, High Range and Low Range
 - Description
 - Custom command
 - Global Alias Browser for expressions and connections
- Support for text objects (text, button, etc.)
 - Label
 - Global Alias Browser for label and descriptions
- Support for display buttons
 - Display file (supported automatically by inheritance from PICK)
 - Label (supported automatically by inheritance from Text)
 - Global Alias Browser for labels and descriptions
- Support for state fields
 - StateValue
 - StateString
 - Default Value
- Access the Global Alias Browser from the description and language edit boxes for Analog Selector, Animator, Color, ColorAnalog, and Digital Selector
- Flash, Hide, Location, Pick, Process Points, Rotation, Size, Time/Date
- State Fields and connected the code behind the pages.
 - Added browse buttons to Local Variables - InitVal, LoRange, HiRange, DisplayTabs, Windows Properties
- Display Properties
 - Caption
 - Download Value 2x
 - Toggle Value 3x
- New pick action Select GAS (Global Aliasing System) Theme
- Command Line Support: GraphWorX allows you to set the initial global alias themes using a command line argument:
`-Themes="MACHINE=<Buildings=Building1"`
The syntax conforms to pick action Select GAS (Global Aliasing System) Theme syntax.

For more information about global aliasing, please see the Global Aliasing Configurator Help documentation.

Section 3

Managing Display Files

Introduction

This section explains how to manage your GraphWorX display files. The File menu contains the functions that allow you to create, open, save, and print display files.

File Menu

The **File** menu commands are listed in the table below.

File Menu Commands

Command	Shortcut Keys	Function
New	CTRL+N	Creates a new display file.
Open	CTRL+O	Opens an existing display file.
Save	CTRL+S	Save the current display file.
Save As		Saves the display as a specified file type or version type.
Print		Prints out the current display.
Print Preview		Shows you how a display will look on the printout before printing the display.
Print Setup		Configures the printer settings.
Exit		Closes the application.

GraphWorX File Name Extensions

In GraphWorX, a file can be saved as a:

- **GraphWorX display file (*.gdf) (with or without Visual Basic for Applications)**
- **GraphWorX template file (*.tdf)**
- **GraphWorX display file for Windows CE (*.gdc)**
- **GraphWorX template file for Windows CE (*.tdc)**
- **Symbol file, or *.sdf**

Working with GraphWorX Display Files

New



Select **New** from the **File** menu to create a new GraphWorX display file in your application.

Open



Select **Open** from the **File** menu to open an existing GraphWorX display file.

Save As

To save the GraphWorX display as a specified file type or version type, select **Save As** from the **File** menu. This opens the **Save As** dialog box, as shown below.

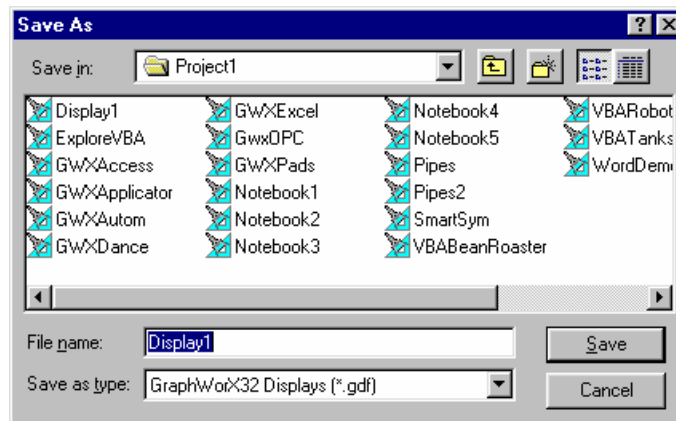


Figure 3.1. Save As Dialog Box

The following options allow you to specify the name and location of the file you're about to save:

File Name

Type a new file name to save a document with a different name. A file name can contain up to eight characters and an extension of up to three characters. Your application adds the extension you specify in the **Save As Type** box.

Save As Type

The **Save As Type** box allows you to save as .gdf (normal) or .tdf (template). If you select an existing display name, it overwrites the display.

Note: Each display will be saved as non-VBA until you execute a VBA-related command.

Save



Select **Save** from the **File** menu to save a GraphWorX display file.

Save As Previous Version

The **Save As Previous Version** feature enhances the compatibility of current GraphWorX displays with previous versions of GraphWorX. GraphWorX so far supported the backward compatibility, which means that the older display could be loaded into a newer version of GraphWorX. The Save As Previous Version feature allows you to save a display file created in a newer version of GraphWorX in one of the previous versions while preserving the existing version of the display file, so that the display file can be still loaded in the previous version of GraphWorX in which it was created.

Opening a Display File

When opening a display file, GraphWorX detects the current display version. If an earlier version of the display was detected, GraphWorX displays the following message, as shown in the figure below. You have two options: either update the display to the latest version of GraphWorX or keep the earlier version of the display.

- **To update the display to the latest version, click the Yes button. All the new features of the latest version can be used and stored in the display. Such a display can be then loaded only in the latest version of GraphWorX.**
- **To force GraphWorX to keep the earlier version of the display, click the No button. The next time you save the file it will be saved in that version, and then it can be loaded in a GraphWorX of that version or a newer version.**

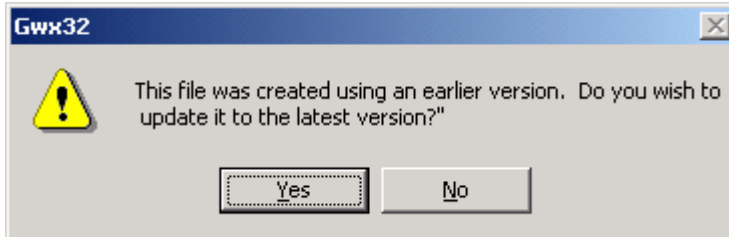


Figure 3.2. Message Indicating That the Display File Was Created Using an Earlier Version of GraphWorX

Saving a Display File

When you save a display file, GraphWorX checks to see what features were added to the display and if those features are supported in the GraphWorX version that is currently selected. The current version is selected based on the loaded display file, and the version can be changed explicitly by using **File > Save As** dialog box.

If the version is the latest version, or if no unsupported features were added to the display file, GraphWorX saves the changes normally, without any warning message. If unsupported features were added to the display file, GraphWorX informs you about it and gives you two options, as shown in the figure below:

- **To update the version to the latest file format and save all the new changes, click the Update Version button. (Such version cannot be loaded in the version of GraphWorX that was used to create the display)**
- **To preserve the current display version and remove the items that made the display file incompatible with the current file version (i.e. features that did not exist in this version of GraphWorX), click the Remove Incompatible Items button. By removing the incompatible items, the display is saved in the earlier version and therefore it can be loaded into that version and any later version of GraphWorX.**

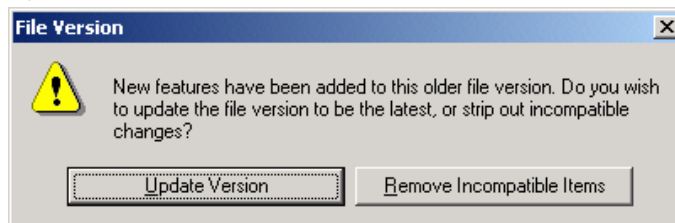


Figure 3.3. Option To Update Version of File or To Remove Incompatible Features

Saving a Display File As a Specific Version

To save a display file explicitly in one of the supported previous versions of GraphWorX:

1. Select **Save As...** from the File menu in GraphWorX.
2. In the **Save As** dialog box, select a version of GraphWorX from the **Save as type** drop-down list, as shown in the figure below.
3. Click the **Save** button. The display is saved in the selected version of GraphWorX. The version information for the file is preserved internally so that the next time you save the file by selecting **Save** from the File menu, the file will be resaved in the selected version.

Note: The display can be saved either with VBA or without VBA. The non-VBA version is required by the GraphWorX Viewer ActiveX control and non-VBA GraphWorX.

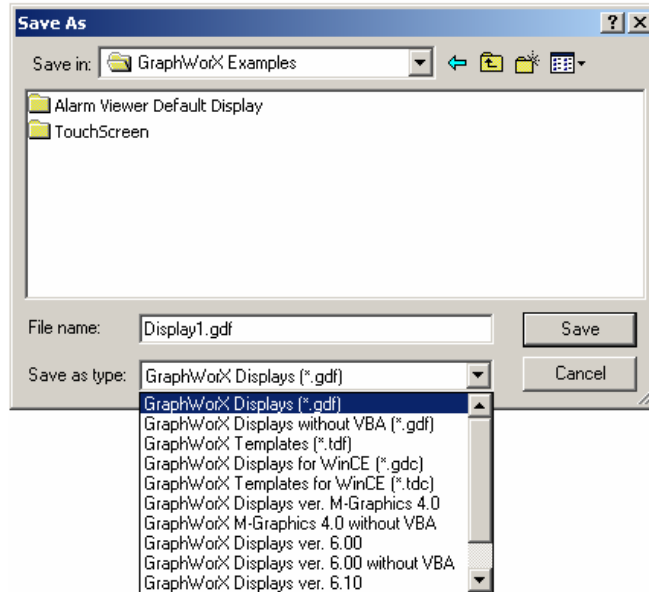


Figure 3.4. Saving a Display File as a Previous Version of GraphWorX

Loading a Display in Runtime

When a display is loaded in runtime mode, the version information messages are suppressed but do not stop the runtime processing. However, when GraphWorX is switched to configuration mode, GraphWorX checks to see if the currently loaded display is the latest version or one of the earlier versions, and the message box appears again to let you decide which version to use. You have two options: either update the display to the latest version of GraphWorX or keep the earlier version of the display.

- To update the display to the latest version, click the **Yes** button. All the new features of the latest version can be used and stored in the display. Such a display can be then loaded only in the latest version of GraphWorX.
- To force GraphWorX to keep the earlier version of the display, click the **No** button. The next time you save the file it will be saved in that version, and then it can be loaded in a GraphWorX of that version or a newer version.

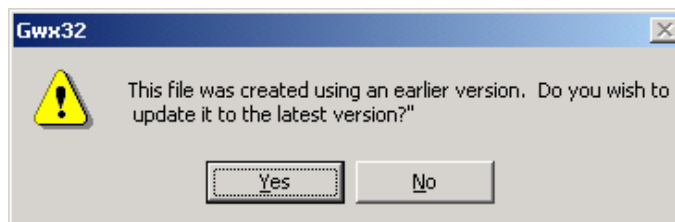


Figure 3.5. Message Indicating That the Display File Was Created Using an Earlier Version of GraphWorX

Suppressing Version Compatibility Information

GraphWorX allows you to suppress the version information messages, working always with the latest version only.

To enable version suppression:

4. **Select Application Preferences from the Format menu in GraphWorX. This opens the Application Preferences dialog box, as shown in the figure below.**
5. **Click on the General tab. Check the Load previous versions without warnings check box. Click OK.**
6. **The value of the check box is automatically stored in the registry and is retrieved when either GraphWorX or this dialog box is loaded again.**

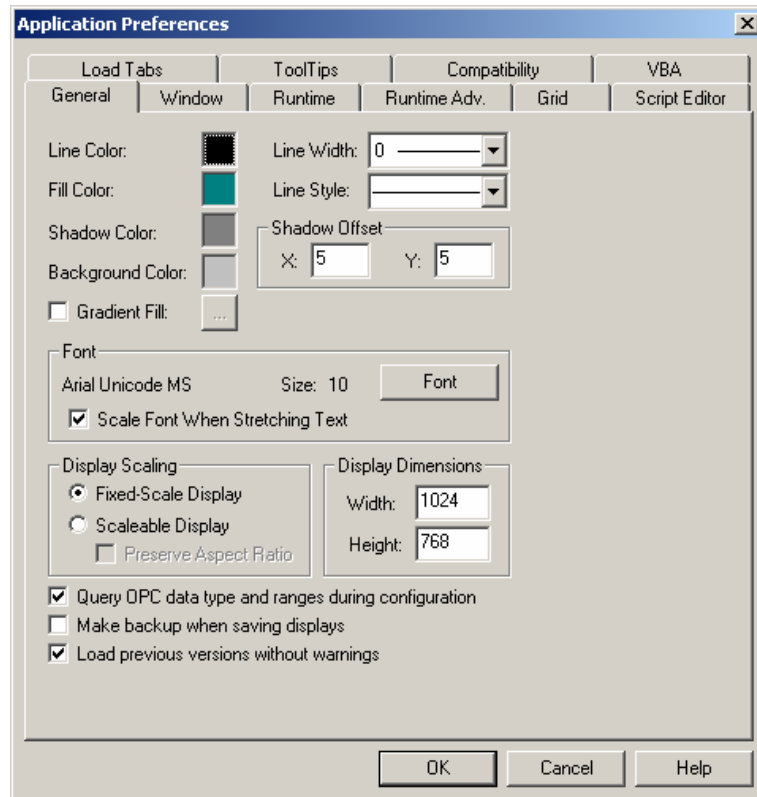


Figure 3.6. Application Preferences: General Tab

Automation Enhancements

When the display is loaded or saved using automation methods, the warning messages are automatically suppressed so they do not interrupt the script. Three new automation methods allow full control over the file versioning in automation:

- Function `GetFileVersion()` As long
- Function `GetFileContainsVBA()` As Boolean
- Function `SetFileVersion(NewFileVersion As long) As long`

The function **GetFileVersion()** returns the following values:

- -1 .. `OpenFileError (&HFFFFFF)`
- 0 .. `Ver520`
- 1 .. `Ver600`
- 2 .. `Ver600SP`
- 3 .. `VerMGraphics40`
- 4 .. `Ver610`
- 9999 .. `VerLatest (&H270F)`

The function **GetFileContainsVBA()** returns True value if the display currently contains the VBA part and False if the display does not contain the VBA part. The presence of the VBA part is controlled by loading a non-VBA display and saving the display as non-VBA, or by using the **FileSaveAsType(1)** (contains VBA) or **FileSaveAsType(0)** (does not contain VBA) automation methods.

The function **SetFileVersion()** sets a display version for the next save. Note that the set of versions supported for saving is less than the set of versions detected on reading the file. In other words, GraphWorX detects more versions of files, but it allows saving only in the more recent versions; version 5.20 and older versions are not supported. If there is an attempt to save in an unsupported older version, then GraphWorX saves in the version SetVer600.

SetFileVersion() input values

- 2 .. SetVer600SP
- 3 .. SetVerMGraphics40
- 4 .. SetVer610
- 9999 .. SetVerLatest (&H270F)

The automation interface is very powerful, and it can be used for automated display processing (e.g. there is a need to upgrade or downgrade a group of displays to a certain version, or to make certain changes in the displays while preserving the existing version).

Visual Basic Code Sample

```
Private Sub DisplayVersion()
    ' Specify the path and filename of an existing display
    Const strPath As String = "C:\PathToYourDisplay\"
    Const strFilename As String = strPath & "YourDisplay.gdf"
    Const SetVer600SP as Long = 2
    Const VerLatest as Long = 9999

    ' Create GWX instance
    Dim g As New Gwx32.GwxDisplay

    ' Check if GWX was created
    If g Is Nothing Then
        MsgBox "Failed to create GWX"
        Exit Sub
    End If

    ' Show GWX in the front and load the requested display
    Call g.ShowWindow
    Call g.BringWindowToTop
    Call g.FileNew
    Call g.FileOpen(strFilename)

    ' Check the file version
    Dim ver As long
    ver = GetFileVersion(g)

    ' If this is the latest version, save in the
    ' version 6.00
    If (ver = VerLatest) Then
        g.SetFileVersion(SetVer600SP)
        g.FileSave
    End If

    ' Exit GWX, the FileNew suppresses the message box
    ' if to save changes.
    g.FileNew
    g.ExitApplication
End Sub
```

```
' Get file version and trace the current version in the
' Immediate window for debugging purposes.
Private Function GetFileVersion(g As GwxDisplay) As Long
    Const OpenFileError as Long = -1
    Const Ver520 as Long = 0
    Const Ver600 as Long = 2
    Const Ver600SP as Long = 2
    Const VerMGraphics40 as Long = 2
    Const Ver610 as Long = 2
    Const VerLatest as Long = 9999

    Dim ver As long

    ' Check the version
    ver = g.GetFileVersion
    Select Case (ver)
        Case OpenFileError:
            Debug.Print "Version: FileOpenError"
        Case Ver520:
            Debug.Print "Version: Ver520"
        Case Ver600:
            Debug.Print "Version: Ver600"
        Case Ver600SP:
            Debug.Print "Version: Ver600SP"
        Case VerMGraphics40:
            Debug.Print "Version: VerMGraphics40"
        Case Ver610:
            Debug.Print "Version: Ver610"
        Case VerLatest:
            Debug.Print "Version: VerLatest"
    End Select

    ' Check if the display contains VBA
    If g.GetFileContainsVBA Then
        Debug.Print "VBA: Yes"
    Else
        Debug.Print "VBA: No"
    End If

    GetFileVersion = ver
End Function
```

Known Limitations

Version Detection

The limitation is detecting the version of a display that contains only static objects and ActiveX controls but no dynamic objects. GraphWorX detects the version of the display based on the version of the most advanced dynamic object used in the display. If there are only static objects or ActiveX controls in the display, GraphWorX is not able to detect which version of the display this is and considers it to be the latest version.

Static Objects and ActiveX Controls

Static objects did not change between the GraphWorX 6.0 Service Pack and the 6.10 versions, so such a display can be still open in the previous version of GraphWorX.

ActiveX controls, however, keep the version within itself and GraphWorX does not have any control over it. The consequence is that **a display with embedded TrendWorX and AlarmWorX ActiveX controls fails to load properly in version 6.00 SP if it was saved in the latest version of GWX.**

Explanation of the Problem

In the case of ActiveX controls, GraphWorX asks each ActiveX control to serialize its content into the current display, but GraphWorX has no control over what is being serialized by each control. If a new version of the control is installed, it will apparently save in the latest version.

Printing GraphWorX Displays



To print a GraphWorX display, select **Print** from the **File** menu, or click the **Print** button on the **Standard** toolbar. This command opens the **Print** dialog box, where you may specify the range of pages to be printed, the number of copies, the destination printer, and other printer setup options.

- **Printer:** This is the active printer and printer connection.
- **Name:** Select a printer from the list of available printers.
- **Properties:** Choose the Properties option to change the page layout and paper size.
- **Print Range:** Specifies the number and range of pages to print out.
- **Copies:** Specifies the number of copies to print out.

Shortcuts:

Toolbar:



Keys:

CTRL+P

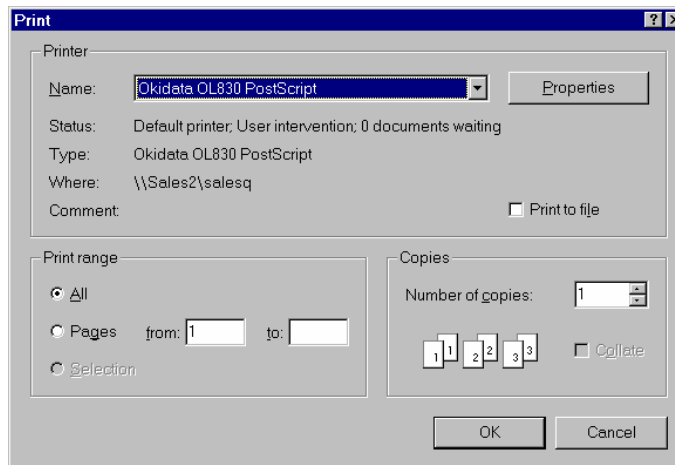


Figure 3.7. Print Dialog Box

Print Setup

Selecting **Print Setup** from the **File** menu opens the **Print Setup** dialog box, which allows you to select a printer and printer connection. You can also specify the **Paper Size**, **Source**, and **Orientation**, as well as select the destination printer and its connection.

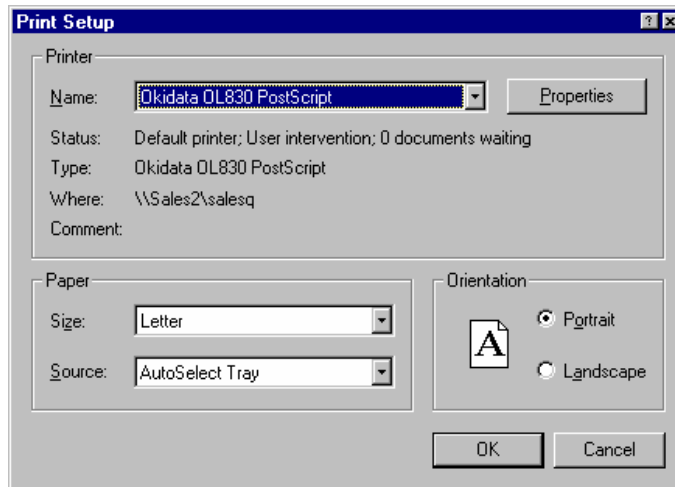


Figure 3.8. Print Set Up Dialog Box

Summary Information

Selecting **Summary Info** from the **View** menu opens the **Summary Information** dialog box, shown below.

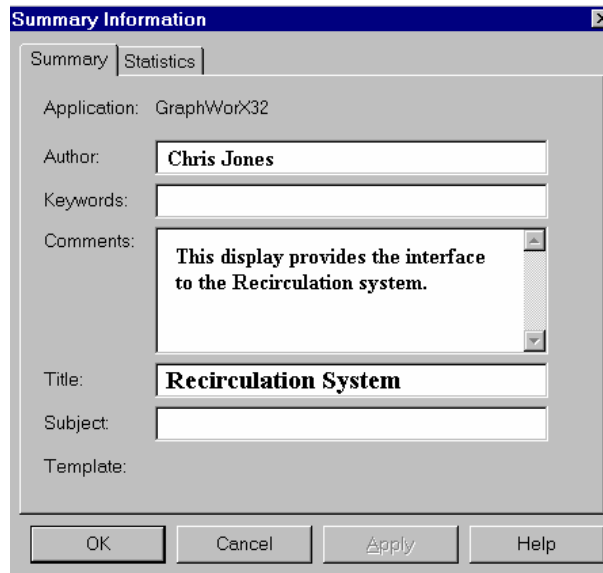


Figure 3.9. Summary Information Dialog Box

Summary Tab

In the **Summary** tab, you can fill in the details such as **Author**, **Keywords**, **Comments**, **Title**, and **Subject** in the spaces provided.

Statistics Tab

The **Statistics** tab, shown below, provides general information about the file currently on display.

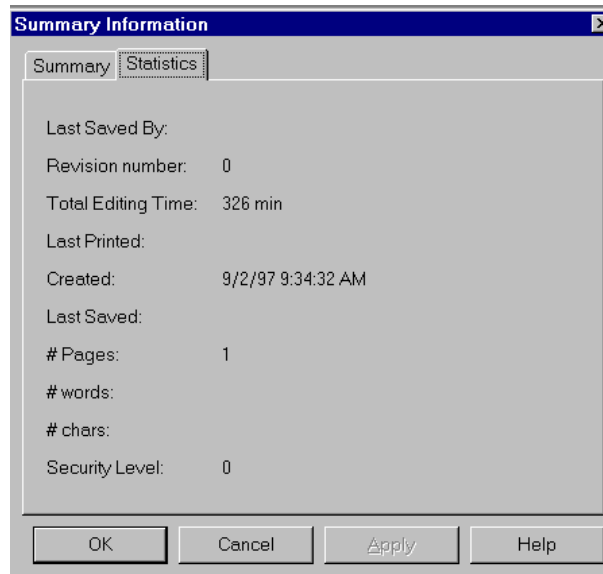


Figure 3.10. Statistics Tab

Note: Summary Information for a display can also be viewed in Windows Explorer. In Windows Explorer, right-click the display file and choose **Properties** from the pop-up menu. The **Summary** and **Statistics** tabs will appear in the dialog displayed by Windows Explorer.

Exiting GraphWorX

To close GraphWorX, select **Exit** from the **File** menu. You can also use the **Close** command on the application **Control** menu. You will be prompted to save documents with any unsaved changes.

Shortcuts:

Mouse: Double-click the application's Control menu button.

Keys: **ALT+F4**

Note: If you have not saved the changes in configuration mode and you exit during runtime mode, your changes will not be saved.

Templates

Templates are standard formats with established parameters that can be applied to a display at any time in the configuration mode.

GraphWorX supports two file extensions: .gdf (graphic display file), default file extension and .tdf (template display file).

The following functions are provided under the **Template** submenu of the **Format** menu.

Apply Template

A template file can be applied to a display at any time in configuration mode. Select **Template > Apply Template** from the **Format** menu. Select a template (.tdf) file and then click **Open**. The specified template is loaded into the current display. Any objects already in the display are preserved. Also, the current summary information for the display is preserved. If you apply a template to a display that already has a template applied, the old template will be replaced by the new one. Objects in the applied template cannot normally be edited (they are essentially frozen background elements), however these objects can be modified using the "Edit Applied Template" functionality described below.

Remove Applied Template

To remove all the objects in a previously applied template, select **Template > Remove Applied Template** from the **Format** menu.

Edit Applied Template

To edit the objects of an applied template, select **Template > Edit Applied Template** from the **Format** menu. When you enter this mode, only the objects in the template are visible (all other objects in the display are invisible and do not interfere with editing the template). Any edits made to the template's objects only affect the current display, not other displays that use the same template. To perform global edits on a template, see "Update Template Displays" below.

Exit Edit Applied Template

To cancel out of Edit Applied Template mode, select **Template > Exit Edit Applied Template** from the **Format** menu. You can also press the **Escape** key to exit Edit Applied Template mode.

Update Template Displays

To globally update displays that have the same template file applied as the one currently being edited, select **Template > Update Template Displays** from the **Format** menu. This function is only available when editing a .tdf file. You can edit a .tdf file and use this function to apply the changes to the displays that are based on that .tdf file.

Insert Template Objects

Selecting **Template > Insert Template Object** from the **Format** menu opens the **Choose Template Object Type** dialog box, shown below. Template objects are essentially placeholder frames that can later be replaced by a desired object. There are currently three types of template objects: image (e.g. bitmap, jpeg, etc.), metafile, and OLE object. When a template object is added to a display, it shows a text message like "Double Click Here to Add Image." (You can change this message.) When you double-click on the object, you are prompted for an image file that will replace the template object at its current size and location. This feature can be used to lay out generic displays that can be specialized later by filling in the placeholders for images, metafiles, and OLE objects.

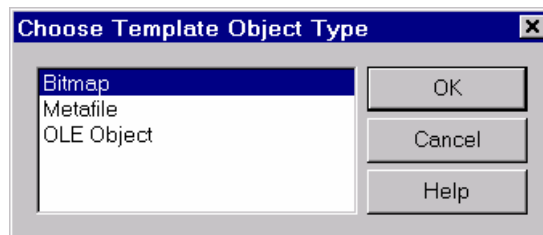


Figure 3.11. Choosing the Template Object Type

Layers

The use of layers in GraphWorX is helpful because it allows you to categorize elements of the display and separate it into levels of detail through decluttering. Using the **Layers** command in the **Format** menu, you can add, remove, or duplicate a layer. You can also edit the layer properties, set the active layer, set the next layer, set the previous layer, hide layers above the current layer and hide layers below the current layer.

Configuring Layers

Every GraphWorX display starts off as a display with one layer, the **primary** or system layer.

Note: This primary layer is the only layer to which a template can be applied.

It is important to understand the significance of this primary layer. When dealing with a display that has several layers, whatever you put in the primary layer appears in all layer views. While the objects placed in the system layer can be seen in the configuration of all other layers, it is not possible to edit these components while editing other layers. To add another layer, select **Layers > Add Layer** from the **Format** menu. This opens the **Edit Layer Properties** dialog box, shown below.

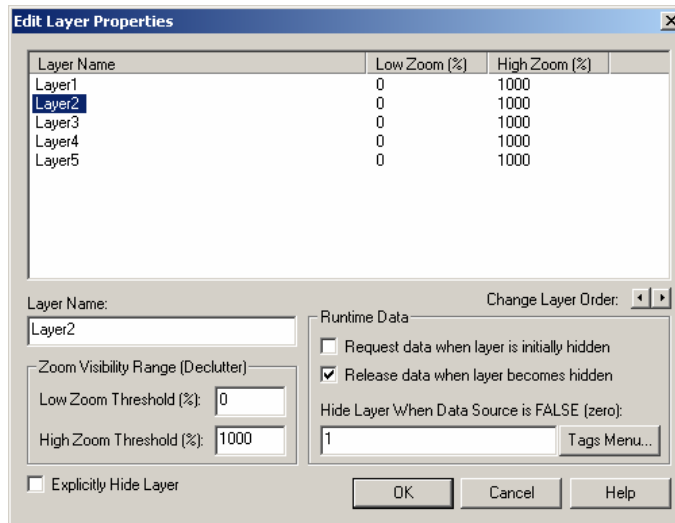


Figure 3.12. Edit Layer Properties Dialog Box

Through this dialog box, you can enter a layer name (this name should indicate what the layer contains or why it is a separate layer and should be unique to this display). Additionally, you can set a number of other configuration elements, including the zoom threshold and the runtime data acquisition elements.

Note: You can select an object in an inactive layer by pressing the **ALT** key and double-clicking the object.

Note: It is possible to change the properties once you have set them by selecting **Layers > Edit Layer Properties** from the **Format** menu.

Zoom Visibility Range (Declutter)

By assigning each layer a different zoom threshold, decluttering can occur. Decluttering involves automatically hiding or showing elements depending on the current zoom status. Zoom thresholds cannot be applied to the primary/system layer. The default values for zoom threshold are 0-1000, which makes the layer always visible. By changing this threshold for certain layers, the zoomed-out view will only show the most basic display.

Runtime Data

The runtime data section of the **Edit Layer Properties** dialog allows you to set how data should be retrieved from the display during runtime. If a layer has never been visible, you have two options: (1) to request data only when the layer becomes visible; or (2) to request data even while the layer is hidden so when the layer becomes visible it will contain valid data. If a layer was at one point visible and is no longer visible, you have the option of continuing to request data while the layer is hidden to maintain valid data or to release the data when the layer is hidden and request data when it becomes visible again.

Checking **Explicitly Hide Layer** hides the layer in runtime regardless of the **Hide Layer When Data Source Is FALSE (zero)** value. The **Explicitly Hide Layer** feature forces the layer to remain hidden regardless of whether the layer would be visible based on the zoom threshold. The layer will remain hidden until this option is turned off, at which point the layer may remain hidden depending on the attached tag value and zoom threshold.

The **Tags Menu** button allows you to set each layer to hide when certain conditions become false (zero). This is helpful when dealing with a multilayer display that is only useful if you can see the active layer. The following options are available from the **Tags Menu** button.

OPC tags. Selecting **OPC Tags** from the **Tags Menu** allows you to select an OPC Tag from the **OPC Universal Tag Browser**, shown below. The tag browser shows all available tags from your machine as well as the network. Once you have selected the appropriate tag, click **OK** to return to the **Edit Layer Properties** dialog box.

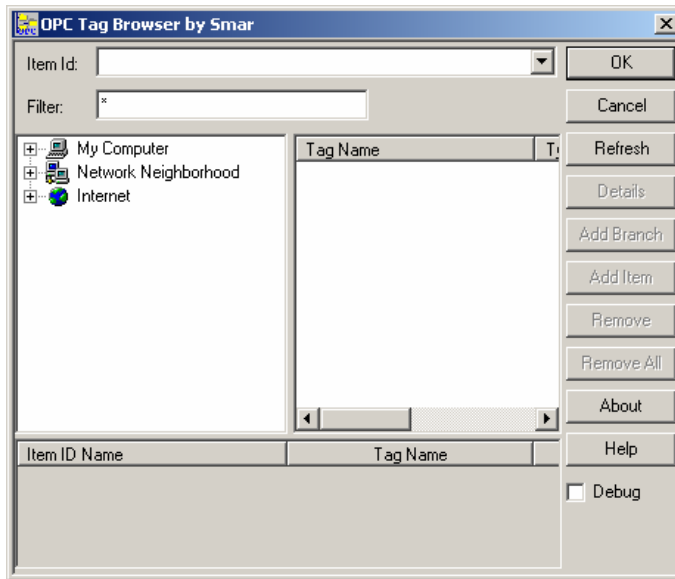


Figure 3.13. OPC Universal Tag Browser

Expression editor. Selecting **Expression Editor** from the **Tags Menu** opens the **Edit Expression** dialog box, shown below. The **Expression Editor** is the same as the expression compiler used in many other areas of GraphWorX. You can use the Expression Editor to create equations, which may or may not contain OPC Tags. Once the equation is set, click **OK** to return to the **Edit Layer Properties** dialog box.

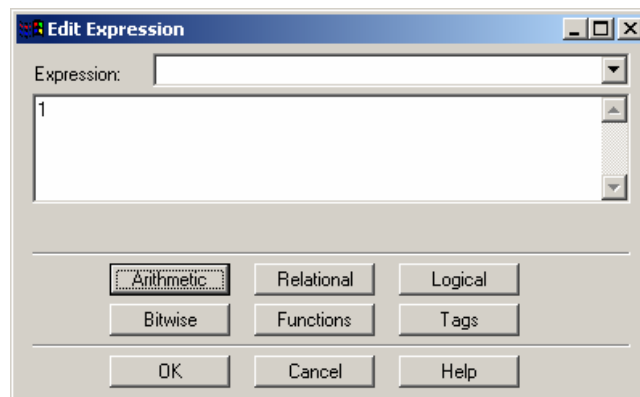


Figure 3.14. Expression Editor

Local Aliases. Selecting **Local Aliases** from the **Tags Menu** opens the following dialog box, which contains all aliases in the current display.

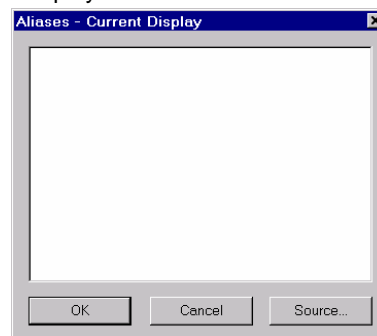


Figure 3.15. Aliases Dialog Box

Note: You may recall that aliases are set by entering an alias name between '<< >>'.

It is also possible to find aliases from other displays by selecting the **Source** button. From this button, two options become available: **Current Display** or **Other Display**. If you select **Other Display**, the **Choose Display** dialog box opens, and then you can navigate throughout the system and network to find the appropriate files. Once you have selected the proper alias, click **OK** to return to the **Edit Layer Properties** dialog box.

Local Variables. Selecting **Local Variables** from the **Tags Menu** opens the dialog box shown below. As with aliases, local variables are created when you are setting up the configuration of different components in a display. Local variables are denoted by the following syntax: `~~localvariable~~`. If this option is selected the following dialog box will be displayed:

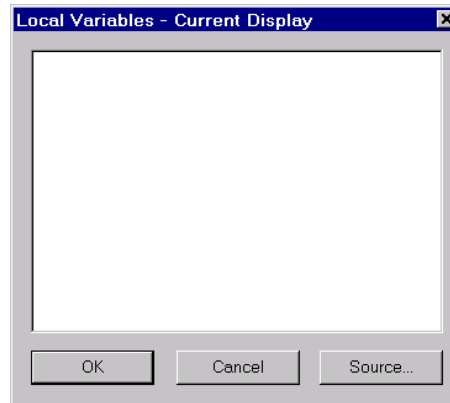


Figure 3.16. Local Variables Dialog Box

The **Source** button is very similar to that of the alias selector box in that you are given the options of **Current Display** or **Other Display**. Once you have selected the appropriate local variable, click **OK** to return to the **Edit Layer Properties** dialog box.

Simulation Variables. Selecting **Simulation Variables** from the **Tags Menu** opens the following dialog box. This contains all variables available from the simulated OPC server.

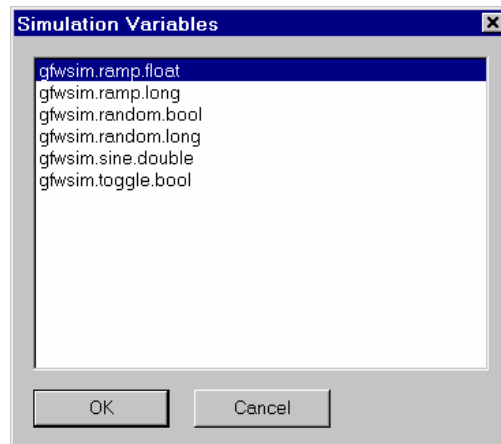


Figure 3.17. Local Variables Dialog Box

Select the appropriate simulated variable, and then click **OK** to return to the **Edit Layer Properties** dialog box.

Note: The variable name is made up of three parts:

- **Server name**
- **Tag name**
- **Data type**

Global Aliases. Selecting **Global Aliases** from the **Tags Menu** opens the Global Alias Browser, as shown in the figure below. Select a global alias from the Global Alias Browser, which includes all

global aliases in the global alias database. This eliminates the need to manually type in the alias name. All global aliases that are configured in the Global Alias Engine Configurator are conveniently available to choose from inside the browser. The tree control of the Global Alias Engine Configurator is mimicked in the tree control of the Global Alias Browser. Select a global alias by double-clicking the alias name (e.g. "Floor" in the figure below). The alias name appears at the top of the browser, which automatically adds the <# and #> delimiters to the alias name. Click the **OK** button.

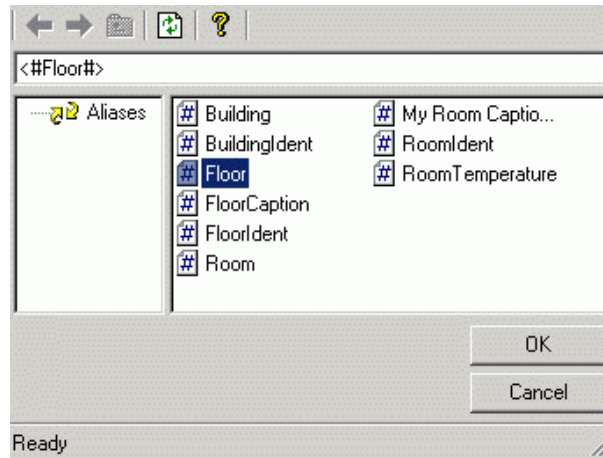


Figure 3.18. Selecting an Alias From the Global Alias Browser

Layers Toolbar

The use of layers in GraphWorX is helpful because it allows you to categorize elements of the display and separate it into levels of detail through decluttering. Using the **Layers** toolbar, shown below, you can add, remove, or duplicate a layer. You can also edit the layer properties, set the active layer, set the next layer, set the previous layer, hide layers above the current layer and hide layers below the current layer. The **Layers** toolbar contains command buttons whose functions are identical to the commands under the **Format > Layers** menu (described above).

The **Layers** toolbar contains 10 numbered buttons (0-9) that allow you to switch between layers, as shown in the figure below. Clicking the **0** button always brings you back to the **Primary Layer**. Each time you create a new layer (in addition to the Primary Layer), a new number button is enabled for that layer. You can add a maximum of 9 layers (buttons **1-9**) that will be indicated in the **Layers** toolbar in addition to the Primary Layer (button **0**). Any additional layers (i.e. more than 9), although present in the display, will not be indicated in the toolbar. Layers are added to the toolbar in the order in which they were created. Hovering the mouse over an active number button displays the layer number in a ToolTip.



Figure 3.19. Layers Toolbar

The **Layers** toolbar includes the following buttons:

- **Add Layer:** Creates a new user layer in the display in addition to the Primary Layer.
- **Remove Layer:** Opens the **Remove Layer** dialog box, allowing you to select which layer to delete from the display.
- **Edit Layer Properties:** Opens the **Edit Layer Properties** dialog box for the currently selected layer in the display.
- **Explicitly Hide Specific Layers:** Opens the **Check Layers to Hide** dialog box, allowing you to select which layer(s) to hide from the display. The **Explicitly Hide Layer** feature hides the layer in runtime and forces the layer to remain hidden regardless of whether the layer would be visible based on the zoom threshold. The layer will remain hidden until this option is turned off, at which point the layer may remain hidden depending on the attached tag value and zoom threshold.

Note: You can also hide a layer by pressing the **CTRL** key and clicking on the layer's numbered toolbar button. For example, if you want to hide layer 7 only, press **CTRL** and simultaneously click on the number 7 button on the **Layers** toolbar.

- **Hide Layers Above Current Layer:** Automatically hides all layers above the currently active layer in the display (configuration mode only).
- **Hide Layers Below Current Layer:** Automatically hides all layers below the currently active layer in the display (configuration mode only).
- **Set Active Layer Previous:** Sets the editable layer to the layer that is just before the currently displayed layer. For example, if the current layer is Layer 3, the editable layer will be set to Layer 2.
- **Set Active Layer Next:** Sets the editable layer to the layer that is just after the currently displayed layer. For example, if the current layer is Layer 1, the editable layer will be set to Layer 2.

Editing Tools

There are two main editing tools that should be noted for the configuration of GraphWorX displays containing layers. The first is the **Hide Layers** feature in the **View** menu. The other is the **View** toolbar, shown below. The first option is also available in runtime mode and is discussed in greater detail in the following section.



Figure 3.20. View Toolbar

Using the **View** toolbar, it is possible to change the view of the display by activating different zoom functions. Additionally, the toolbar provides specific shortcuts for layering. All of the following functions can also be found by selecting **Layers** from the **Format** menu.

Set Active Layer Previous

Selecting **Layers > Set Active Layer Previous** from the **Format** menu sets the editable layer to the layer that is just before the currently displayed layer. For example, if the current layer is Layer 3, the editable layer will be set to Layer 2.

Note: To change the order of the layers, select **Layers > Edit Layer Properties** from the **Format** menu and use the **Change Layer Order** arrow buttons located underneath the layer list box. The active layer name is displayed in the right corner of the status bar.

Set Active Layer Next

Selecting **Layers > Set Active Layer Next** from the **Format** menu sets the editable layer to the layer that is just after the currently displayed layer. For example, if the current layer is Layer 1, the editable layer will be set to Layer 2.

Set Currently Active Layer

Selecting **Layers > Set Currently Active Layer** from the **Format** menu opens a dialog box listing all possible layers, as shown in the figure below. Select the layer you want to edit, and then click **OK**.

Note: The Currently Active Layer defaults to the layer that was active when the display was last saved.

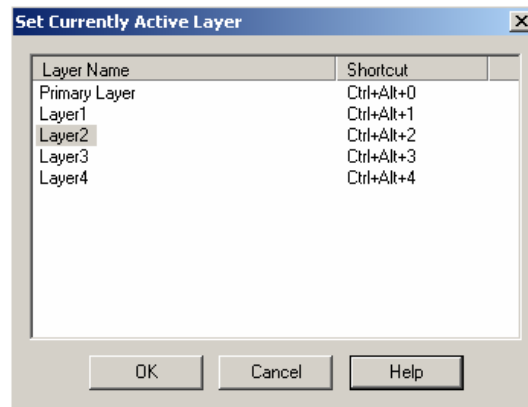


Figure 3.21. Setting the Currently Active Layer

Security for Layers

It is possible to set security for a specific layer within all displays, all layers within a specific display, or all layers in all displays. To set security for a layer, select **Security Configuration** from the **Tools** menu.

This opens the Security Server login dialog box. The format for configuring security for layers is **filename|layername** (ex. **filename<vertical.pipe>layername**).

It is also possible to use "wildcards" when formatting security for layers. The following table provides examples of how to use "wildcards."

Where To Set Security	Example
Specific layer in all displays	*.gdf MyLayer1
All layers in a specific display	MyDisplay1 *
All layers in all displays	*.gdf *

Using Layers in Runtime Mode

You can hide or show layers by selecting **Hide Layers** from the **View** menu or the keyboard shortcut combination **CTRL+SHIFT+0**. Both of these options open the **Check Layers to Hide** dialog box, shown below. Select a layer or layers to hide, and then click **OK**.

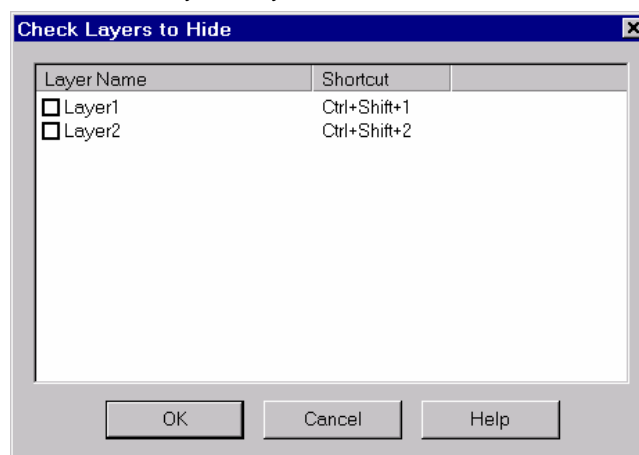


Figure 3.22. Selecting Which Layers to Hide

Layers may also be hidden due to the value of the tag to which they are attached, as well as if you select **Layers > Edit Layer Properties** from the **Format** menu and select the **Explicitly Hide Layer** check box in the **Edit Layer Properties** dialog box, as shown in the figure below. The **Explicitly Hide Layer** feature forces the layer to remain hidden regardless of whether the layer would be visible based on the zoom threshold. The layer will remain hidden until this option is turned

off, at which point the layer may remain hidden depending on the attached tag value and zoom threshold.

Each layer that you added during configuration is shown in the dialog box, along with the system generated shortcut key. Note that there is a check box to the left of each layer name. If this box is checked, that layer will be hidden during runtime mode.

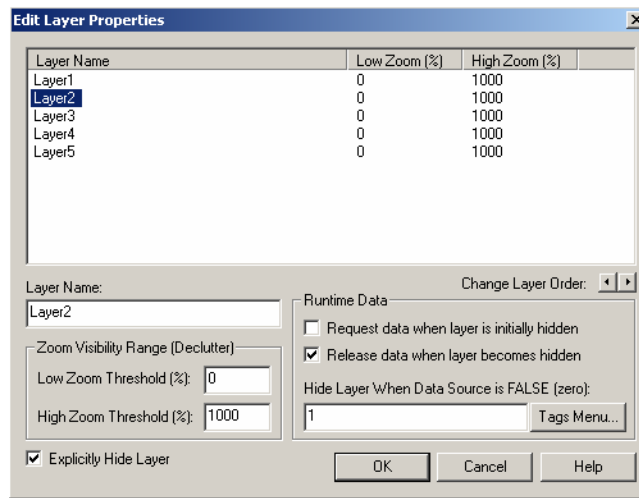


Figure 3.23. Explicitly Hide Layers Function in Layer Properties

Additional VBA Events for Layering

PreAnimateLayer(BSTR layerName)	Fired before data for the layer are requested.
PostAnimateLayer(BSTR layerName)	Fired after data for the layer have been requested.
PreDeanimateLayer(BSTR layerName)	Fired before data for the layer are released.
PostDeanimateLayer(BSTR layerName)	Fired after data for the layer have been requested.

Section 4

Creating and Modifying Objects

Introduction

This section explains how to create and configure objects to use in your GraphWorX displays. You can draw objects, such as circles, boxes, and ellipses using the drawing tools available from the **Draw** menu. You can also define various formats for these objects, including line style and weight, text fonts, custom colors and background colors, using the **Format** menu.

Configuration Mode

Configuration mode is the mode in which you design displays. In this mode, you can create static and dynamic objects, set general display properties, etc. Static objects are typically created on screen and then modified via the property inspector (some static objects, such as polylines, arcs, and text, also have additional on-screen editing capabilities). Dynamic objects are also configured via the Property Inspector.

Display Configuration Mode Password

It is possible password-protect the configuration of a GraphWorX display. To do so, select **Set Configuration Mode Password** from the **Format** menu. This opens the **Set Password for Current Display** dialog box, as shown in the figure below.

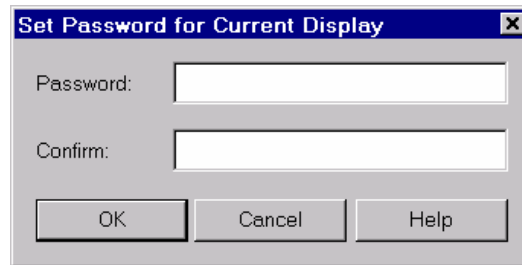


Figure 4.1. Set Configuration Mode Password

Enter your password in the **Password** field, and then re-enter it in the **Confirm** field. When a user tries to open a password-protected display, that user will be presented with the following dialog before being able to enter the configuration.

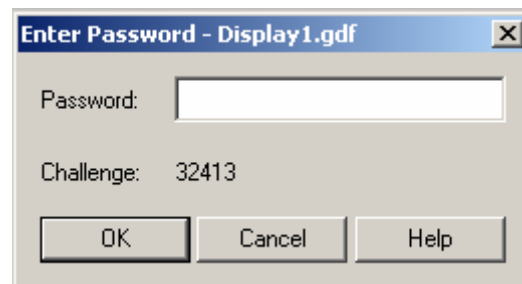


Figure 4.2. Opening a Password-Protected Display

If the correct password is entered, the display opens in configuration mode. If an incorrect password is entered, a warning message is displayed, and the user will continue to be denied access until the proper password is entered.

Note: The display password does not apply to displays opened in runtime mode.

Arrow Key Resize

While in configuration mode, you can change the size and shape of an object using the arrow keys along with the **SHIFT** or **CTRL+SHIFT** keys. Holding down the shift key and pressing the arrow keys adjust the top and right sides of the object. Holding down the **CTRL+SHIFT** key combination and pressing the arrow keys adjust the bottom and left sides of the object.

TAB Key Object Selection

You can use the **TAB** key to move the selector from one object to another. The selector will move around the display in the order in which the objects were created, or, if you have placed the objects in a particular order (i.e. brought an object forward or sent one back), then the selector moves from the object farthest (or the selected object) forward.

ALT Key Group Selection

If you press the **ALT** key and try to select a single object, you will notice that nothing happens. This is because the **ALT** key forces you to use the group select function. For the group select function, the select function is active and you draw a rectangle encasing at least one object, in essence selecting all objects entirely enclosed in the rectangle.

Note: An object is selected when the selector handles are visible around the edges of the object.

CTRL+SHIFT Drag

When you press **CTRL+SHIFT** and drag an object, the object will not only be copied, but you will only be able to drag the object in a horizontal or vertical path from the source location. This is helpful when you are trying to line objects up in the same axis.

Template Activity

When a template is changed and saved, you are automatically asked if you wish to have all displays using said template to be updated as well. This is a very effective way to incorporate changes that may affect several displays. Although this may take some foresight on your part, effective planning can save you a lot of time in the long run.

Find and Replace Features

The GraphWorX **Find and Replace** function supports searches based on **Custom Data**, **Description** and **File Name** (for pick dynamics).

Customize Toolbars

You can add, remove, or relocate items on the various toolbars within GraphWorX. Simply right-click on the toolbar you wish to edit. This opens the **Customize Toolbar** dialog box, as shown in the figure below.

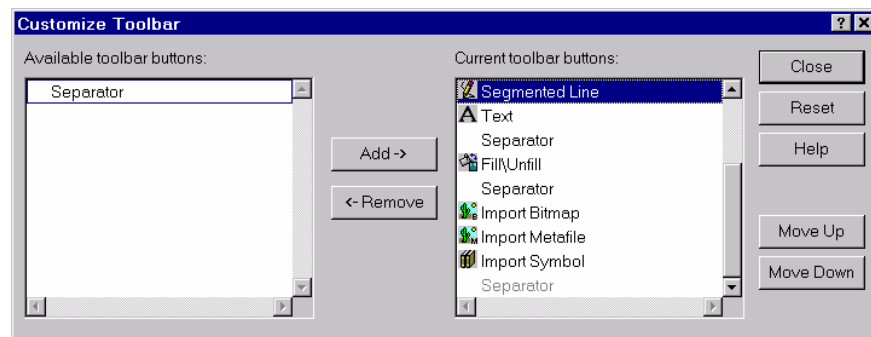


Figure 4.3. Customizing Toolbars

To remove a button from the toolbar, select the button from the list on the right under **Current toolbar buttons** and click the **<-Remove** button. Similarly, if you wish to add a button to the toolbar, select the button from the list on the left under **Available toolbar buttons** and click the **Add->** button. Use the **Move Up** and **Move Down** buttons to relocate buttons on the toolbar. This feature is helpful since there may be functions represented on a toolbar that you do not use (in which case it should be removed) or the default ordering of the buttons does not make any sense to you.

Note: All functions pertaining to a particular toolbar will be in either the list of available buttons or in the list of toolbar buttons. It is impossible to entirely remove a function from the dialog.

Draw Functions

The **Draw** functions enable you to create display objects using various drawing tools. You can create complex drawings by grouping different display objects. The Draw functions are located in the **Draw** menu and the **Draw** toolbar, as shown in the figures below. The information on the bottom left of your screen corresponds to the Draw toolbar.

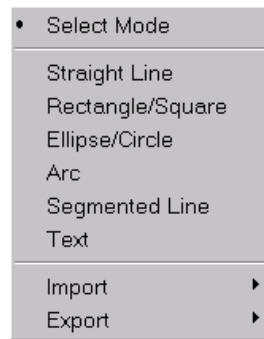


Figure 4.4. Draw Menu



Figure 4.5. Draw Toolbar

Selector Tool



With the **Selector** button, you can choose objects to modify. When you select an object, you can change its appearance or location. Before you can modify an object, you must select it. Once it is selected you can format and edit that object.

To select a single object:

1. Click the **Selector** button on the **Draw** toolbar or right-click the work area to access the **Selector** tool from a list of options. The mouse pointer appears as an arrow.
2. Select an object by placing the pointer on the object and clicking the left mouse button. Square handles surround the object.

To select multiple objects:

1. Click the **Selector** button on the **Draw** toolbar.
2. Draw a box to surround all the objects by pressing the left mouse button and dragging the pointer around the objects, or by pressing the **SHIFT** key and clicking the individual objects.
3. Release the mouse button. Square handles surround each object selected.

Note: The **Select** mode is always checked by default on the **Draw** menu.

Right-clicking the selected object opens a special version of the **Format** menu, as shown below, which is discussed later in this section:

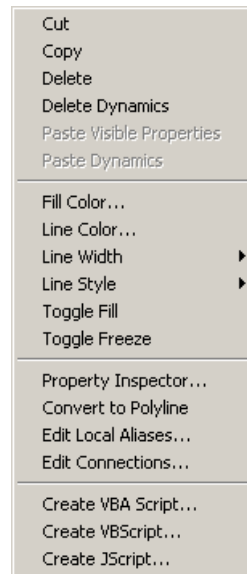


Figure 4.6. Format Menu

The Property Inspector

The **Property Inspector**, shown below, is a property sheet containing a collection of property pages, or tabs. The Property Inspector dialog box lets you view and make changes to object parameters. The primary page is the General Page, which reflects the properties of a given static object (in the figure shown below, a rectangle).

The Property Inspector may also display a variable number of additional pages that show information about attached dynamics. These additional pages are easily accessed via the tabs at the top of the dialog box.

The Property Inspector has four buttons on its Property Sheet: **OK**, **Cancel**, **Apply**, and **Help**. The **OK** button accepts all changes made on all of the property pages and exits the dialog. The **Cancel** button discards all changes made on all of the property pages and exits the dialog box. The **Apply** button accepts all changes made on all of the property pages but does not exit the dialog (subsequently canceling from the Property Inspector will not undo applied changes since those changes have already been accepted).

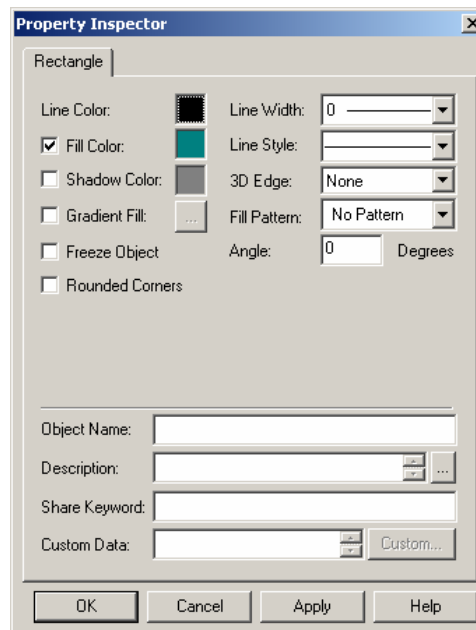


Figure 4.7. Property Inspector Dialog Box

In the **Description** field, you can select an alias to use by clicking the ... button to the right of the text box and selecting either **Global Aliases** or **Language Aliases** from the pop-up menu, as shown in the figure below. This opens the respective alias browsers, allowing you to choose from a list of available aliases.

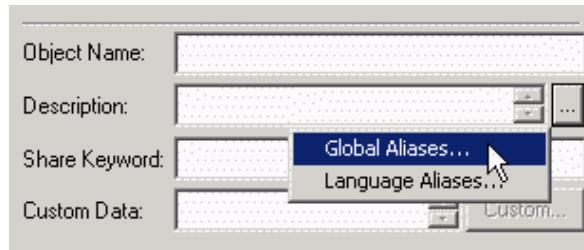


Figure 4.8. Selecting an Alias for the Description Field

Checking and Altering the Parameters of an Object

To check and alter the parameters of an object:

1. Double-click the object. This opens the **Property Inspector** dialog box.
2. Edit the parameters described in the table below, and then click **OK** to return to the work area.

Note: The fields of the **Property Inspector** dialog box vary slightly depending on what type of object is being inspected.

Property Inspector Parameters

Parameter	Description
Line Color	Current line color of the object. Click this field to display the Color Palette and select a line color from the options available.
Fill Color	Current fill color of the object. Click this field to display the Color Palette and select another color from the options available.
Shadow Color	Current shadow color. Click this field to display the Color Palette and select a shadow color for the object from the options available.
Gradient Fill	Clicking the '...' button allows you to set the Gradient Fill options for the object.
Freeze Object	Checking or unchecking the box enables you to enable or disable the ability to move or stretch this object.
Rounded Corners	Rounds out the corners of a rectangle.
Line Width	Current Line Width. Change the width by selecting from the list box.
Line Style	Current Line Style of the object. Change the style by selecting from the list box.
3D Edge	Gives the selected object the 3D edge specified from the options available.
Fill Pattern	Fills the selected object with the chosen pattern from the available options.
Angle	Specifies the angular orientation of the object.
Object Name	Identifies the object for OLE Automation.
Description	Describes the object and usually coincides with what is displayed under ToolTips. Note: You can also select either Global Aliases or Language Aliases from the alias browsers by clicking the ... button to the right of the Description field.
Share Keyword	Used to identify the object for Update Shared Objects.
Custom Data	Allows you to enter custom data.

Gradient Fill

The **Gradient Fill** feature in the Property Inspector is supported for display background, rectangles/rounded-rectangles, ellipses, pies/chords, lines and text - back - fill (including process points, buttons and times/dates). To access the gradient feature, you must first insert one of the previously mentioned items in the display and then double-click the item to open the Property Inspector.

Gradient Configuration Features

It is possible to change the color of the object from the **Configure Gradient** dialog box, shown below. To do so, check the **Gradient Fill** check box in the Property Inspector and then click the ... button. Click on the color box in the **Configure Gradient** dialog box to open the **Color Palette**.

It is also possible to reverse the color (when using both single and two color gradients) by checking the **Reverse Colors** check box.

Note: The color of the gradient object can be changed using the Color Palette.

Additionally, the **Configure Gradient** dialog box contains a preview window that shows what the gradient fill looks like as you adjust the gradient properties.

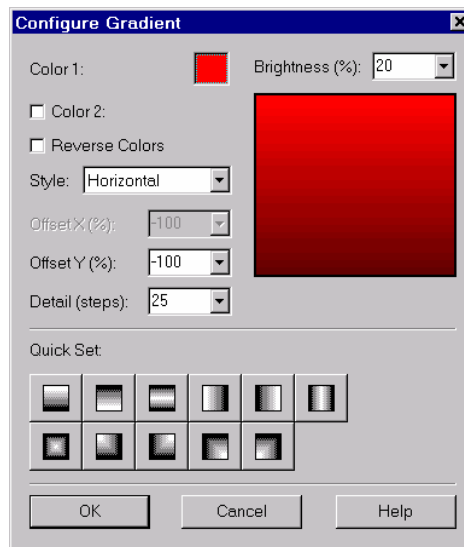


Figure 4.9. Configuring Gradients

Adding Dynamic Actions

When you add a dynamic action to an object that has a gradient, the object acts as it would normally, but the coloration now contains the gradient fill. However, notice that if the object is given a rotation dynamic, the shading of the gradient does not stay in the same place at all times (i.e., as the object rotates past the x-axis, the shading flips sides of the object).

Gradient Styles

There are three different gradient styles: **Horizontal**, **Square** and **Vertical**.

Style: Horizontal		The Horizontal style has the light focus coming from the top or the bottom of the object.
Style: Square		The Square style has the light focus coming from a corner.
Style: Vertical		The Vertical style has the light focus coming from either the right or the left side of the object.

Offsets

The **Offset** feature in the **Configure Gradient** dialog box sets how far off center the light focus is from both the x-axis and y-axis. Two options are available: **Offset X** and **Offset Y**.

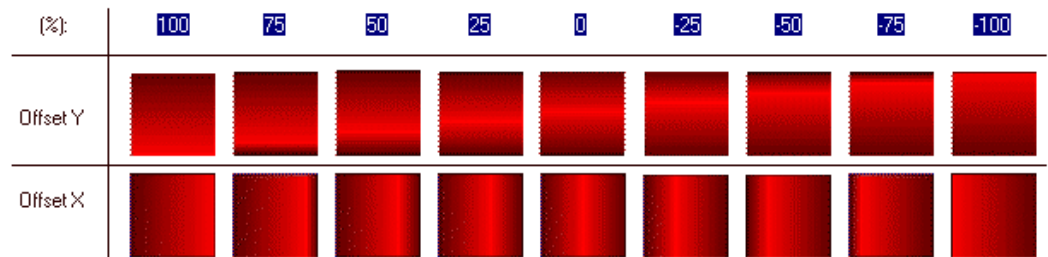


Figure 4.10. X and Y Offsets by Percentage

The figure shown above is an example of nine different light sources for both the Offset X and the Offset Y.

Note: It is possible to enter any number between 100 and -100 for both the X and Y offsets.

Brightness

If you are only using one color, you can select a percentage of brightness. The following is an example of some of the different percentages of brightness that are possible through the **Configure Gradient** dialog box. You can enter any number between 0 and 100 for the brightness level.

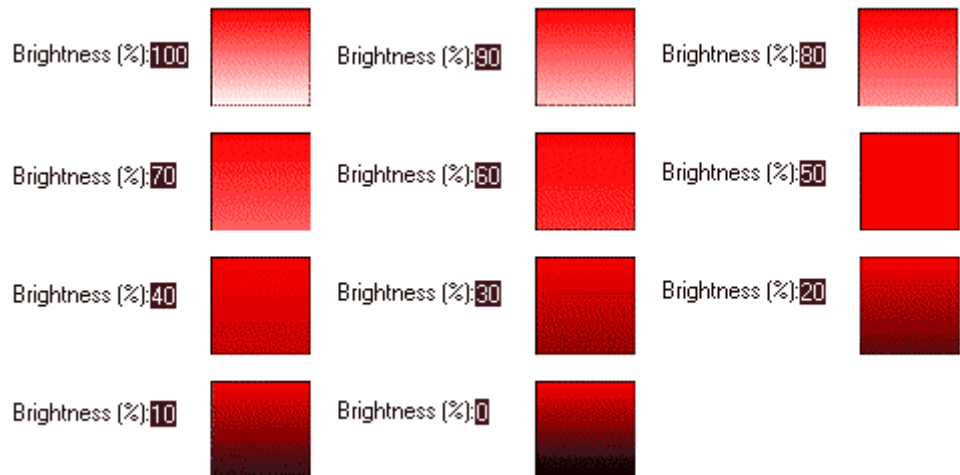


Figure 4.11. Gradient Brightness by Percentage

If you are using two colors, the brightness adjustment option disappears and you are instead allowed to select the second color. This second color is tied to the line color of the shape, even if the second color option is not selected. For example, if you change the second color to yellow and then decide to switch back to a one-color gradient, the border line of the object will still be yellow.

Detail Steps

The **Detail (Steps)** feature in the **Configure Gradient** dialog box sets the smoothness of the gradient's appearance. The following is an example of how some of the different detail steps appear:

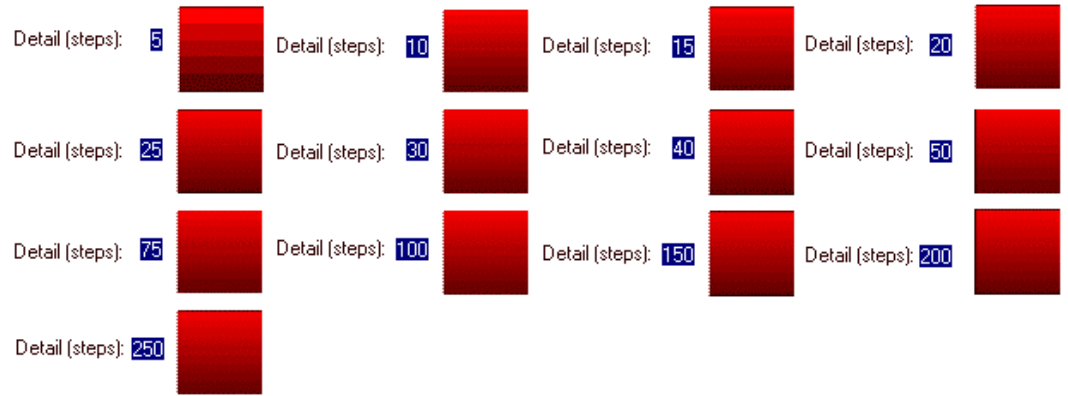


Figure 4.12. Detail Steps for Gradients

Quick Set Buttons

The **Configure Gradient** dialog box also contains the following **Quick Set** buttons. Each quick set button has a predefined configuration, which will override any settings you have already defined for a particular object.

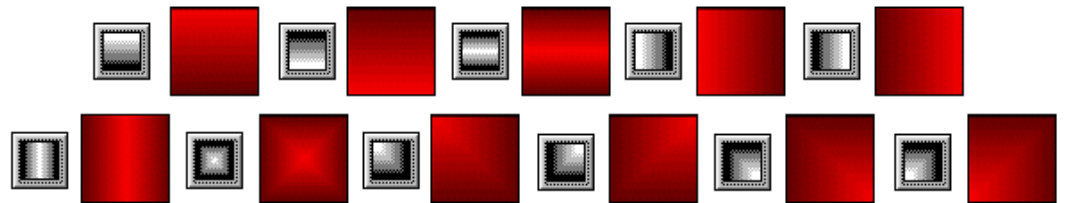


Figure 4.13. Gradient Quick Set Buttons

Lines



The **Line** tool on the **Draw** toolbar enables you to draw straight lines in the work area. You can also define the line style (solid or segmented) and line weight (thickness) using the **Line Style** and **Line Weight** functions. Refer to the Formatting section of this topic for more information.

To draw a line:

1. Select **Straight Line** from the **Draw** menu, or click the **Straight Line** button on the **Draw** toolbar. The mouse pointer appears as a pencil.
2. Pressing **SHIFT** while you draw constrains the line to 0, 45 or 90 degrees.

Double-clicking on the line displays the Line Property Inspector, shown below. Right-clicking the line displays the **Format** menu.

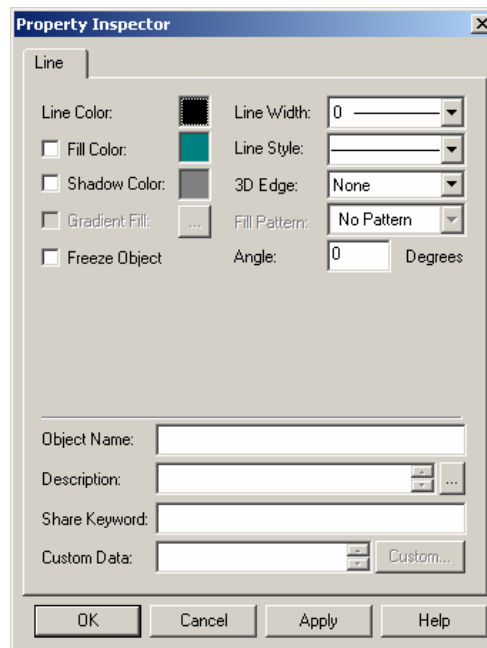


Figure 4.14. Property Inspector: Line

To alter the length of a line:

1. Select the line.
2. Left-click a square handle of the select box, drag it to extend the line, and release.

To reposition a line:

1. Click the line.
2. Drag to a new position.
3. Release mouse button.

To edit the vertices of a line:

1. Select the line.
2. Press **SHIFT** and right-click.
3. The line's vertices appear, and the mouse cursor appears as an upward-pointing arrow. You can now edit the midpoints and vertices of the segments.

Segmented Lines



The **Segmented Line** button on the **Draw** toolbar allows you to draw a flexible, segmented line that can be straight or curved.

Keep the left mouse key pressed for free-hand drawing. Click and release the mouse from point to point. Double-clicking on the line opens the Property Inspector specific to the segmented line, which is identical to the Property Inspector page for the straight line.

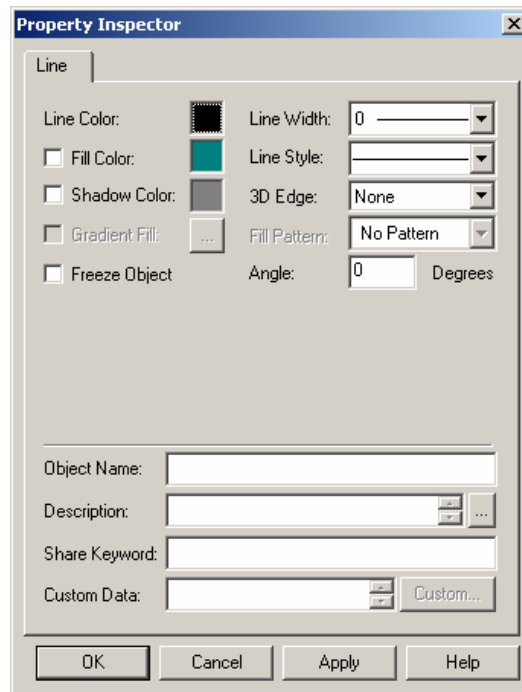


Figure 4.15. Property Inspector: Line

When you create a segmented line, and the final vertex is close to the starting vertex, the polygon will automatically be closed off.

Rectangles



You can use the **Rectangle** tool to draw filled or unfilled rectangles in the work area. The color of the object's border is the currently selected color in the Color Palette.

To draw a rectangle:

1. Select **Rectangle/Square** from the **Draw** menu, or click the **Rectangle/Square** button on the **Draw** toolbar. The mouse pointer changes to a box and cross.
2. Click the left mouse button and drag to form the box. To form a square box, press **SHIFT** while you draw the box.

To view the properties of the box, double-click on the box. The Property Inspector for the rectangle object opens, as shown in the figure below. The Rectangle properties contain the unique check box **Rounded Corners**, which rounds out the corners of the rectangle or square, in addition to the standard property parameters.

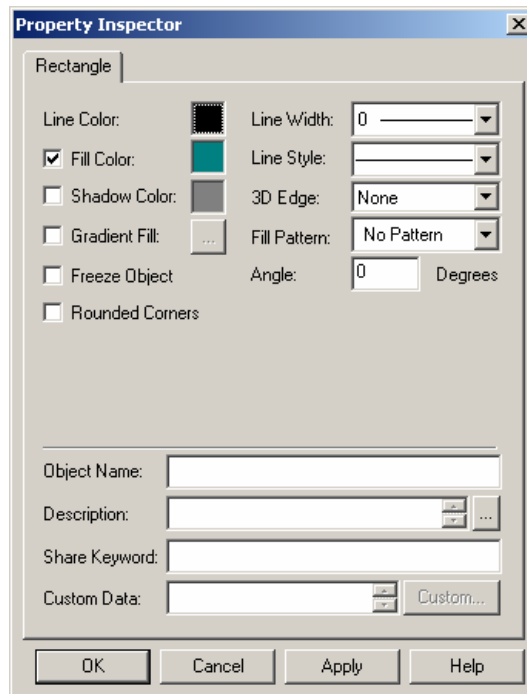


Figure 4.16. Property Inspector: Rectangle

Right-clicking on the rectangle displays the **Format** menu, which provides options for formatting the rectangle. The **Format** menu is discussed later in this section.

Note: The cursor remains in the rectangle mode until you double-click, right-click, press **Esc**. This applies to all drawing operations.

To resize a rectangle:

1. Select the rectangle, and then click the left mouse button on a square handle; drag to resize.
2. Use the **SHIFT** key to maintain the aspect ratio of the rectangle while resizing.

Filling and Unfilling

You can fill and unfill color in rectangles by using the toggle fill function explained in the **Format** section in this topic. You can also change the color by selecting the rectangle and selecting a different color in the **Color Palette**.

Ellipses



With the **Ellipse** tool, you can draw elliptical or circular objects.

To draw an ellipse:

1. Select **Ellipse/Circle** from the **Draw** menu, or click the **Ellipse** in the **Draw** toolbar.
2. Click the left mouse button and drag the pointer anywhere in the work area to form an ellipse. To form a circle, press **SHIFT** while you draw.

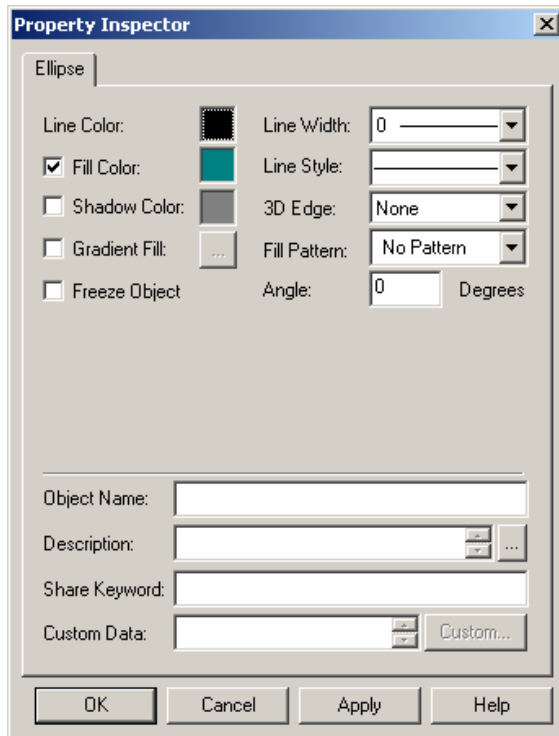


Figure 4.17. Property Inspector: Ellipse

Right-clicking the ellipse displays the **Format** menu

To resize the ellipse:

1. Select the ellipse.
2. Click the left mouse button on a square handle and drag it to resize the ellipse. Press **SHIFT** while resizing to maintain the aspect ratio of the object.

Text



Use the **Text** tool to create text objects.

1. Select **T** from the **Draw** menu. Click the **Text** button on the **Draw** toolbar.
2. Left-click on the work area where you want to place the text object.
3. Type the text you wish to appear and left-click outside the text to insert.

When you right-click on the selected text object, the **Format** menu appears with the **Edit Text** option. Right-click and press **SHIFT** to directly edit the text on screen. Or you can double-click on the object to open the Property Inspector, and then edit the text attributes according to the parameters given in the dialog box, as shown below.

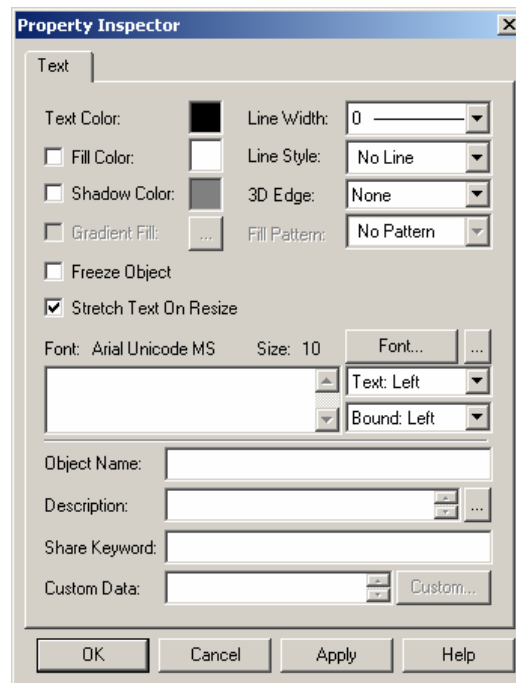


Figure 4.18. Property Inspector: Text Object

You can type a text message directly inside the box under **Font**. You can select an alias to use for the text message by clicking the ... button to the right of the Font button and selecting either **Global Aliases** or **Language Aliases** from the pop-up menu, as shown in the figure below. This opens the respective alias browsers, allowing you to choose from a list of available aliases.

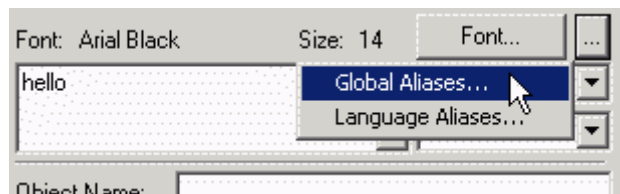


Figure 4.19. Selecting an Alias for the Text Box




The text object Property Inspector parameters are listed in the table below:

Text Property Inspector Parameters

<i>Parameter</i>	<i>Description</i>
Text Color	Specifies a text color from the Color Palette.
Fill Color	Specifies the color of the text background from the Color Palette.
Shadow Color	Specifies the color of the text shadow from the Color Palette.
Gradient Fill	If the Fill Color box is checked, the fill color will have a gradient applied to it.
Freeze Object	Locks the text object in one position.
Stretch Text on Resize	If this option is checked, the text will stretch with the text box when it is resized.
Line Width	Determines the width of the text box.
Line Style	Determines the style of the text box.
3D Edge	Determines the style of the 3D edge.
Fill Pattern	Determines a pattern for the background of the text object.

Font Selection	Description
Text Entry Area	This allows you to edit the text in the text box.
Font	Displays the font used for the selected text.
Size	Displays the point size for the selected text.
Style	Displays style of the selected text.
Font button	Opens the font dialog box to change the parameters for selected text object, such as font, size, and style.
Text: Left, Center, or Right	Aligns the paragraph text to the left, center, or right.
Bound: Left, Center, or Right	Bounds text to the left, center, or right of text box.

The figure shown below provides some examples of how you can configure text to appear in your display:

	Text with no attributes
	Text with fill color and 3D raised edge
	Text with fill color, 3D raised edge and shadow color

Adding Text Attributes

Arcs



You can use the **Arc** tool to draw arc-shaped objects in the work area.

To draw an arc:

1. Select **Arc** from the **Draw** menu, or click the **Arc** button on the **Draw** toolbar.
2. To determine the arc's radius, press and hold the left mouse button, drag the pointer away from the start point, and click.
3. To determine the arc's length, press and hold the left mouse button over one of the boxes on either end of the arc. Drag the arc to the desired length.

To view the properties of the arc, double-click it to bring up the Arc Property Inspector, shown below.

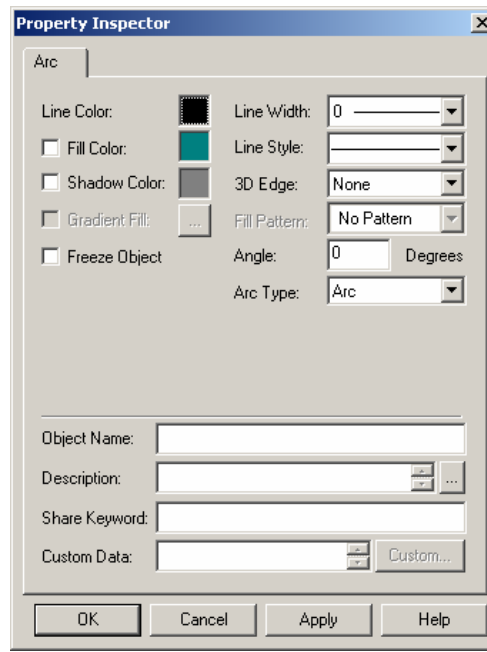

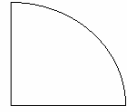



Figure 4.20. Property Inspector: Arc

In addition to the standard parameters, this Property Inspector contains an **Arc Type** selection. The figures below show examples of arc types:

Arc	
Pie	
Chord	

To resize an arc:

1. Select the arc.
2. Press the left mouse button on a square handle, and then drag it to resize the arc and release.

To edit the length of an arc:

1. Right-click the arc to display the **Format** menu, and then choose **Edit Arc**.
2. The mouse pointer changes to a box and cross, allowing you to change the length of the arc.

To convert an arc to a polyline:

1. Select the arc.
2. Right-click the arc to display the **Format** menu, and then choose **Convert to Polyline**.
3. Now you have the option of merging the arc with other line objects to create a single polyline object. This allows you to apply properties such as fill to the whole object. To merge two objects, select both objects and then select **Group Into Symbol** from the **Arrange** menu or toolbar.

Import Functions

Images



By selecting the **Import Image** button on the **Draw** toolbar, or selecting **Import > Image** from the **Draw** menu, you can import an image file (.bmp) into your work area. Images can be resized and moved within a GraphWorX display.

Since the early versions, GraphWorX has been able to import external image files into display files. Until now the import functionality has been limited to the only bitmap (.bmp) image file format. The bitmap format has been in the past years the most common over any Microsoft platform, but today many other graphic formats are available and widely used.

Unfortunately the bitmap format is also the simplest one, and the least efficient in terms of disk space. Some graphic formats (e.g. .jpg and .png) are used because of the minimum disk space that they require, while other formats (e.g. Adobe Photoshop .psd) are used for their flexibility and powerful capability. GraphWorX supports the following image types for import:

- Adobe Photoshop Files (*.PSD)
- C64 Koala Graphics (*.KOA)
- CompuServe GIF (*.GIF)
- Dr. Halo (*.CUT)
- Icon (*.ICO)
- IFF Interleaved Bitmap (*.IFF)
- JPEG (*.JPG;*.JPEG)
- Kodak PhotoCD (*.PCD)
- Liner Bitmap Graphics (*.LBM)
- Multiple Network Graphics (*.MNG)
- Zsoft Paintbrush (*.PCX)
- Portable Network Graphics (*.PNG)
- Portable Network Media (*.PBM;*.PGM;*.PPM)
- Sun Raster Images (*.RAS)
- Truevision Targa (*.TGA;*.TARGA)
- Tag Image File Format (*.TIF;*.TIFF)
- Windows or OS/2 Bitmap (*.BMP)
- Wireless Bitmap (*.WBMP)

To import an image into a GraphWorX display file:

1. Select **Import > Image** from the **Draw** menu in GraphWorX (or click the **Import Image** button on the **Draw** toolbar), as shown in the figure below.

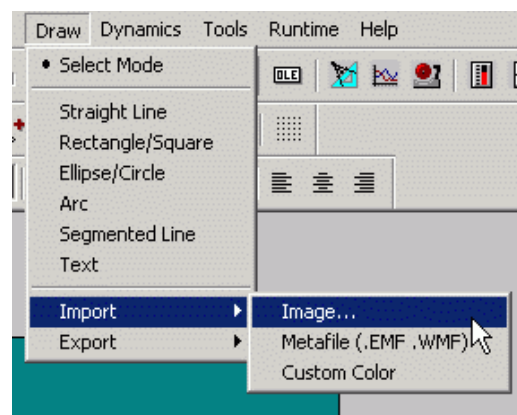


Figure 4.21. Importing an Image File Into a GraphWorX Display File

2. In the **Open** dialog box, select an image file type from the **Files of type** drop-down list, as shown in the figure below. Select an image file to import, and then click **Open**.

3. The image is inserted into the GraphWorX display.

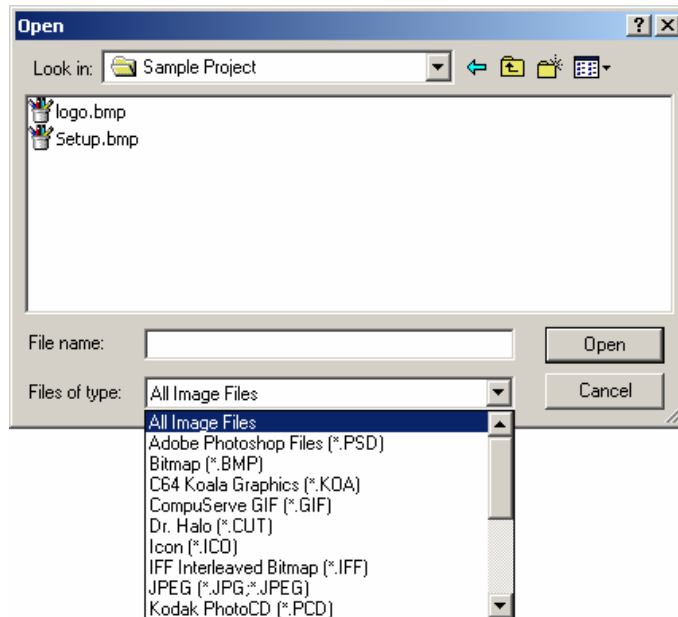


Figure 4.22. Selecting an Image File and Image Type To Import

Changing an Image File

To change an image that has been imported into a GraphWorX display:

1. Right-click the image and choose **Change Image** from the pop-up menu.
2. In the **Open** dialog box, select an image file type from the **Files of type** drop-down list. Select an image file to import, and then click **Open**.

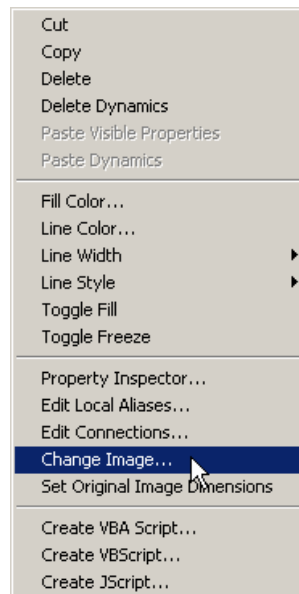


Figure 4.23. Changing an Image

Neural Network / PNG Image Compression

GraphWorX is capable of importing many images formats. Some of these formats use compression (i.e. .jpeg) and others do not (i.e. .bmp). Large images may require a lot of disk space. For example, a single true color bitmap file 1024 X 768 pixels may require up to 2 MB of disk space.

In order to reduce the disk space required from GraphWorX displays, a new compression algorithm has been added. The new image compression can help save disk space and optimize performance when manipulating bitmaps or delivering displays over the Internet through WebHMI.

The new image compression algorithm is based on Neural Network technology and PNG compression. The Neural Network helps reduce the color needed to represent an image, thus reducing the final file size. The PNG compression algorithm saves additional disk space.

The following comparison chart illustrates the differences in file size between standard BMP storage and advanced PNG compression:

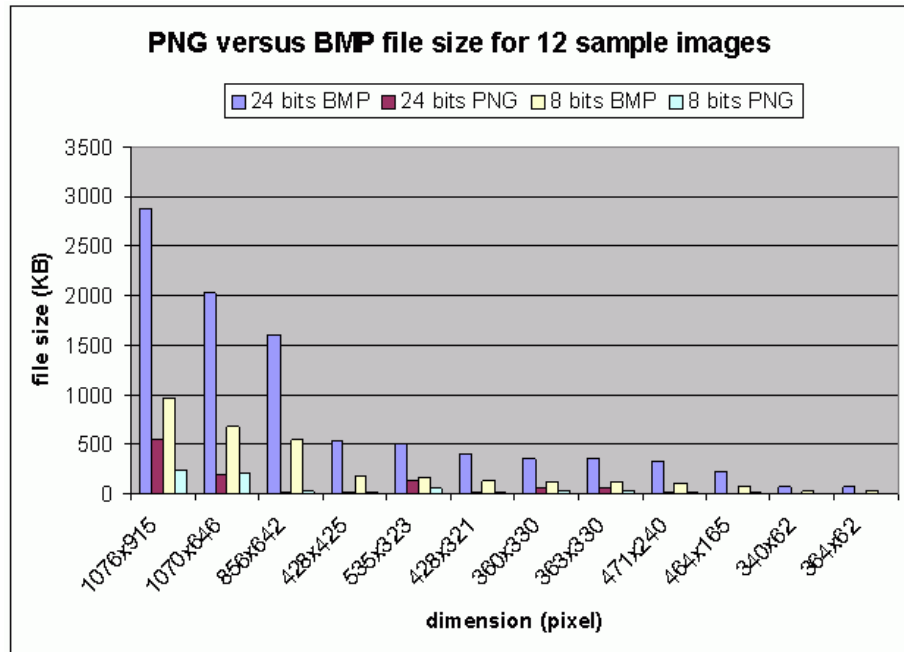
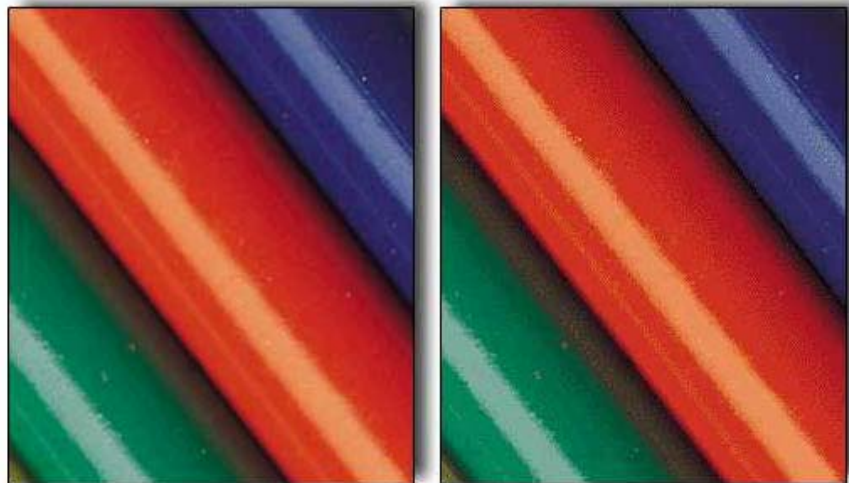


Figure 4.24. Comparison of BMP and PNG Image Formats

The images below show the Neural Network color reduction algorithm. The image on the left uses 16 million colors to represent shades of red green and blue. The image on the right is the result of a first pass through the Neural Network color optimization algorithm. The two images look almost identical, and only a sharp eye can tell the differences between the two, mainly related to the dithering.



Original Image (16 million colors)

Color Reduced Image (256 colors)

Metafiles



By selecting the **Import Metafile** button on the **Draw** toolbar, or by selecting **Import > Metafile** from the **Draw** menu, you can import a windows metafile (.wmf) or an enhanced metafile (.emf) into your work area. As with other images, it is possible to resize and move a metafile within a GraphWorX display.

Custom Colors

By selecting **Import > Custom Color** from the **Draw** menu, you can import a text file that contains red, green, and blue color parameters for the selected object in the display.

Export Functions

Exporting Images and Metafiles

By selecting **Export** from the **Draw** menu, you can export an enhanced metafile (.emf) or an image from your work area.

Sometimes it is useful to have an image that represents a GraphWorX display. GraphWorX provides a tool for exporting a GraphWorX display into an image file. The **Export > Image File** command on the **Draw** menu is available only if something is present inside the display. The resulting image file represents the display content as it is seen on the monitor. The windows and all the toolbars are stripped out of the image. For example, if the original display before export looks like this...

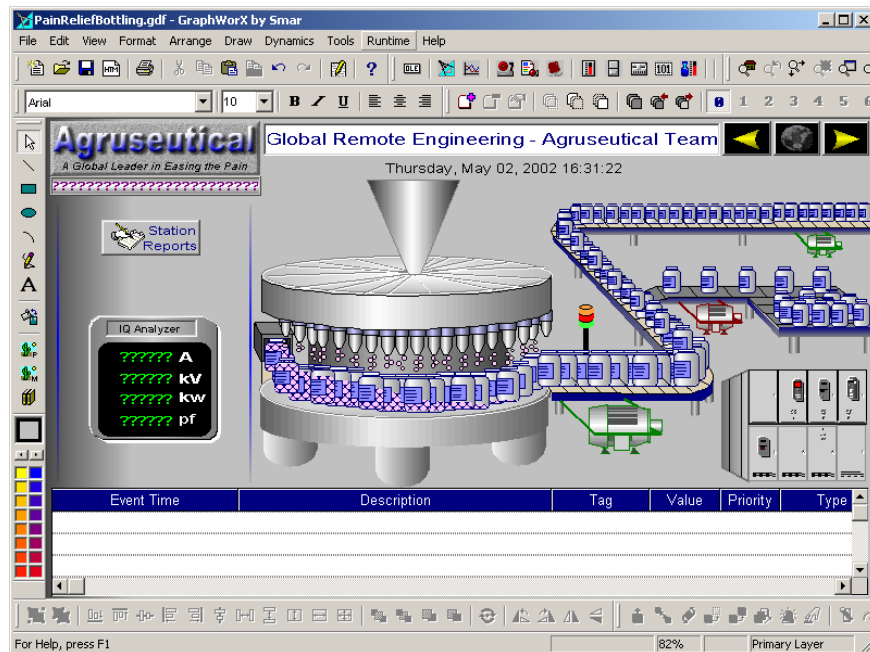


Figure 4.25. Display File Before Export

...then the final image file looks like this:

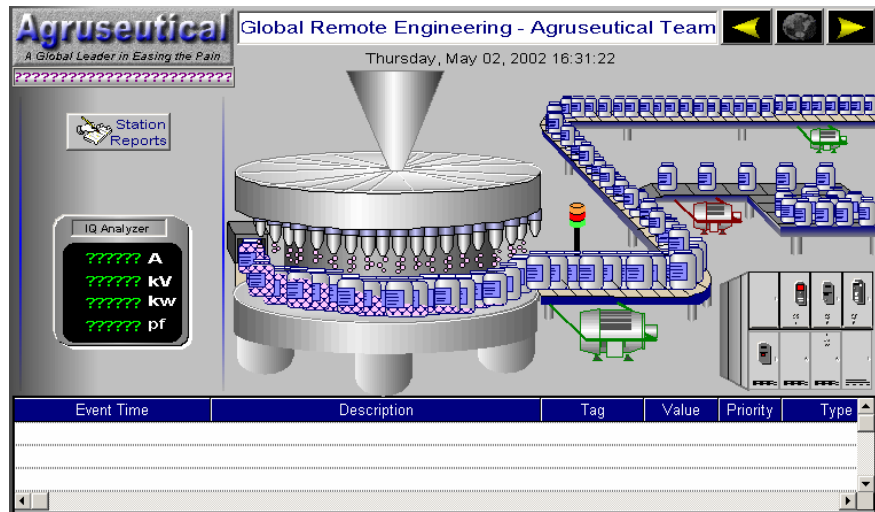


Figure 4.26. Image File After Export

The following image types are supported for export:

- Bitmap (*.bmp)
- Jpeg (*.jpg)
- Png (*.png)
- Tiff (*.tif)

Each format has advantages and drawbacks. The bitmap format, for example, requires more disk space. The jpeg format saves some disk space but also sacrifices some image quality.

To export a GraphWorX display file to an image file:

1. Select **Export > Image File** from the **Draw** menu in GraphWorX, as shown in the figure below.

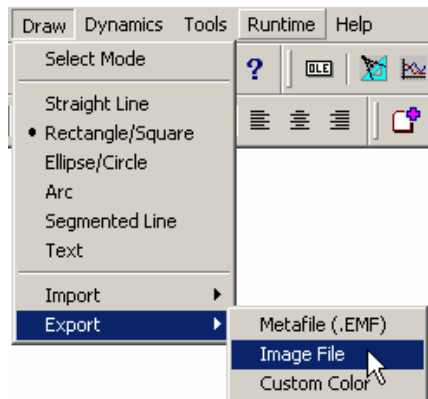


Figure 4.27. Exporting a GraphWorX Display File to an Image File

2. In the **Save As** dialog box, select an image file type from the **Save as type** drop-down list, as shown in the figure below. Give the image file a name, and then click **Save**.

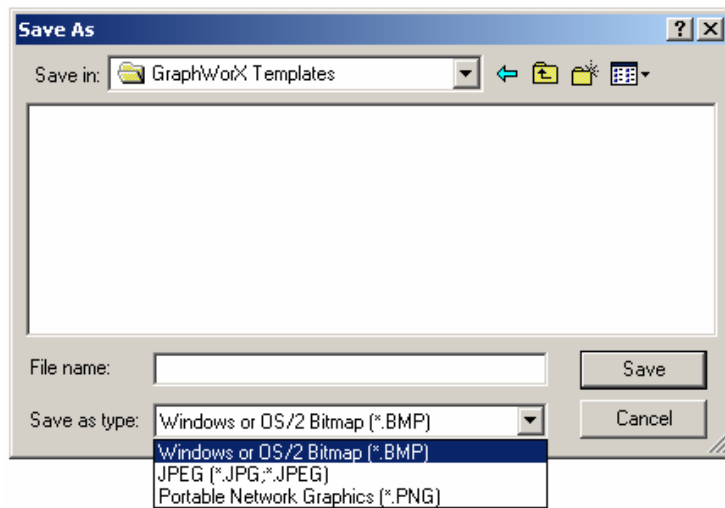


Figure 4.28. Saving the Image File

Exporting Custom Colors

By selecting **Import > Custom Color** from the **Draw** menu, you can import a text file that contains red, green, and blue color parameters for the selected object in the display.

GraphWorX Symbols



GraphWorX uses two different modes for the Symbol Library:

- Standard stand-alone Symbol Library
- Dockable symbol toolbar

To switch between these two modes:

1. Select **Application Preferences** from the **Format** menu.
2. The **Application Preferences** dialog box appears, as shown in the figure below. Select the **Compatibility** tab. Under the **Symbol Library Style** field, you can select **Stand-alone** mode or **Dockable as Symbol Toolbar** mode.

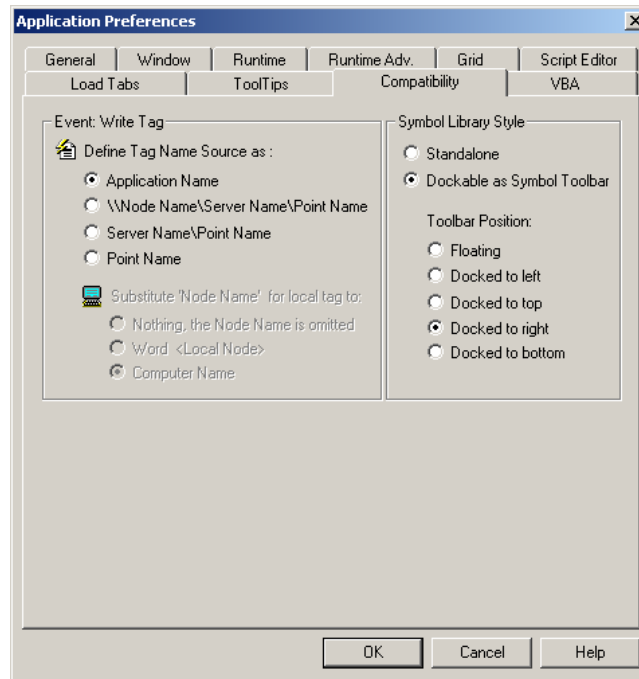


Figure 4.29. Switching Between Stand-alone and Dockable Symbol Modes

Stand-alone Symbol Library

If **Stand-alone** mode is active, clicking the **Symbols** button on the **Draw** toolbar launches the **SymbolLibrary.exe**. You can also select **Import > Symbol** from the **Draw** menu, as shown in the figure below.

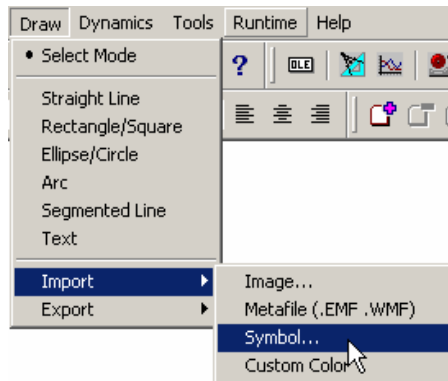


Figure 4.30. Importing Symbols

Dockable Symbol Toolbar

If the **Dockable as Symbol Toolbar** option is selected, then the Symbol Library is replaced with a Symbol toolbar inside the GraphWorX display, as shown in the figure below. The **Symbols** button on the **Draw** toolbar toggles the visibility of the symbol toolbar (instead of launching the Symbol Library). The **Toggle Symbol Toolbar** command on the **View** menu is enabled, and the **Show Toolbars** dialog (which appears when you select **Toolbars** from the **View** menu) also includes a **Symbols** check box for showing/hiding the symbol toolbar. The **Import > Symbol** command is removed from the **Draw** menu.

You can also position the symbol toolbar as:

- A floating window
- Docked to left
- Docked to right
- Docked to bottom
- Docked to top

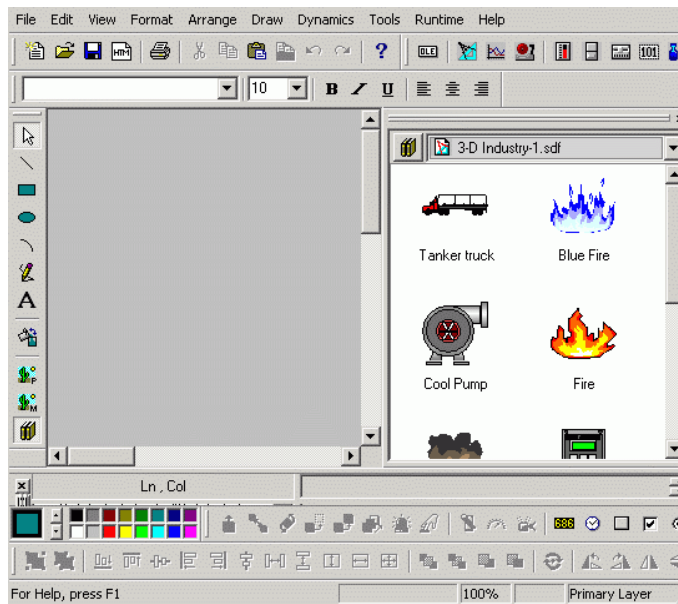


Figure 4.31. Symbol Toolbar Docked Inside a GraphWorX Display

Registry Presettings

The mode for the Symbol Library can be specified during the installation process by creating the following registry entries under

[HKEY_LOCAL_MACHINE\SOFTWARE\ICONICS\Gwx32\Compatibility Properties]:

- **Compatibility Mode:** Can be either Stand-alone or Dockable Symbol Toolbar
- **Dockable Mode:** Symbol toolbar can be a floating window, docked to left, docked to right, docked to bottom, or docked to top

Note: These settings are saved for each user independently in

[HKEY_CURRENT_USER\Software\ICONICS\Gwx32\Compatibility Properties]

Formatting Objects

The **Format** menu provides options for layers, formatting line color, line style, line width, font type, and background color for your display. It also allows you to fill and unfill selected objects, to freeze and unfreeze objects, and to define font style and size.

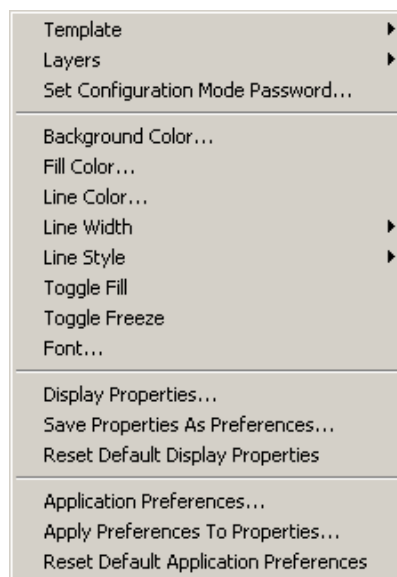


Figure 4.32. Format Menu

Background Color

Selecting **Background Color** from the **Format** menu function defines the color of the background of the selected object from the **Color Palette**, shown below.

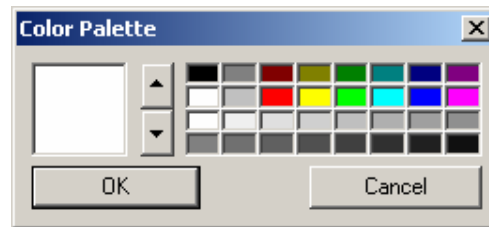


Figure 4.33. Color Palette

Color Palette Menu

Double-clicking on the **Color Palette** displays the **Color** dialog box, shown below. This allows you to choose from a wider range of colors than are available on the basic color palette.

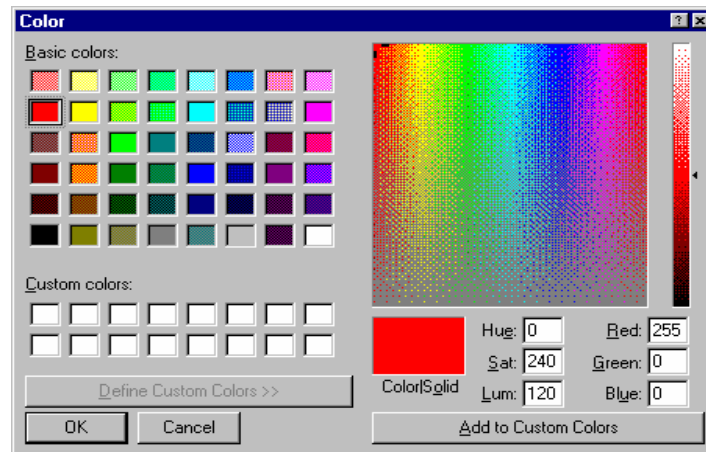


Figure 4.34. Color Dialog

You can use a slider to choose from a range of colors and patterns in the color spectrum at right. The color will be displayed in the **Color/Solid** box. You can then add a custom color or a textured color by clicking the **Add to Custom Colors** button.

Right-clicking the **Color Palette** displays the menu shown in the figure below. When any or all of these parameters are checked on the menu above, the color palette can be kept large when floating, large when docked, viewed with two rows when large, three rows when large, or four rows when large.

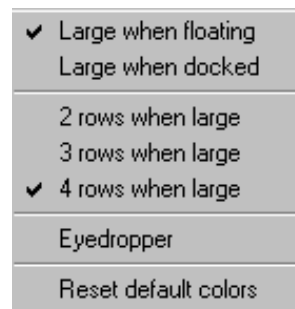


Figure 4.35. Color Palette Menu

Eyedropper

Selecting **Eyedropper** from the **Color Palette** menu allows you to select a color from anywhere in the display and apply it within the display. Click anywhere in the GraphWorX window over the color you want to pick up. Deposit the color in the color palette by clicking over the desired color box in the rows of available colors.

Reset Default Colors

You can restore the default colors by selecting **Reset Default Colors** from the **Color Palette** menu.

Fill Color

Selecting **Fill Color** from the **Format** menu opens the **Color Palette**, from which you can choose a color to fill a selected object in your display.

Line Color

Selecting **Line Color** from the **Format** menu opens the **Color Palette**, from which you can choose a color for a selected line in your display.

Line Style

Selecting **Line Style** from the **Format** menu allows defines how the line appears in a display. Lines can appear as solid or segmented. Currently this works with a line that has no width. There are several line styles from which to choose.

To define a line style:

1. Select the object (line, rectangle, segmented line, ellipse or arc) in your display.
2. Select **Line Style** from the **Format** menu.
3. Select a line type.

Note: Line styles only exist for lines with a width of 0.

Line Width

Selecting **Line Width** from the **Format** menu defines the thickness of a line.

To define the line width:

1. Select an object (line, rectangle, segmented line, ellipse, or arc) in your display.
2. Select **Line Width** from the **Format** menu. The line width options appear to the right of the menu.
3. Select the line width.

Toggle Fill

Selecting **Toggle Fill** from the **Format** menu fills or unfills the inside of an object with the object's current color.

To fill an object with color:

1. Select the object.
2. Select **Toggle Fill** from the **Format** menu or click the **Toggle Fill** button on the **Draw** toolbar.

Note: Lines, bitmaps, and metafile images cannot be filled or unfilled with color.

Toggle Freeze

Selecting **Toggle Freeze** from the **Format** menu disables the movement or stretching of aobjects in the work area. This function is useful for anchoring objects in a display.

To freeze or unfreeze an object:

1. Select the object.
2. Select **Toggle Freeze** from the **Format** menu. A lock appears on the object indicating it is frozen.

You can also freeze an object through the Property Inspector by checking the **Freeze** check box.

Font

Selecting **Font** from the **Format** menu defines the font types, styles, and point sizes for text. You can define font attributes before or after you create text objects.

To define font types:

1. Select the text object(s) in your display.
2. Select **Font** from the **Format** menu to open the **Font** dialog box.

Define the font parameters and click **OK**. The text appears with the font parameters you defined.

Arranging Objects

The **Arrange** menu, shown below, and the **Arrange** toolbar provide you with options for moving objects in layers; grouping and ungrouping objects; rotating, aligning, and evenly spacing selected objects in the display; and making objects the same size.



Figure 4.36. Arrange Menu



Figure 4.37. Arrange Toolbar

Group Into Symbol



The **Group Into Symbol** function on the **Arrange** menu and the **Symbol** button on the **Arrange** toolbar allow you to group several objects into one symbol. The resulting symbol can then be moved as one object.









Ungroup



The **Ungroup Symbol** function on the **Arrange** menu and the **Ungroup Symbol** button on the **Arrange** toolbar allow you to ungroup the symbol into its component objects.

Aligning Objects

The **Align** function on the **Arrange** menu enables you to align objects according to their bottoms, tops, middles, lefts, rights, and centers, as well as to make the objects the same height, the same width, and the same size. The **Align** function on the **Arrange** menu and the **Align** buttons on the **Arrange** toolbar adjust the relative positions of objects in the work area. You can align objects in the following positions:

Bottom		Rights	
Lefts		Centers	
Middles		Even Across	
Tops		Even Down	

To align objects:




1. Select the objects.
2. Select **Align** from the **Arrange** menu, and choose a position or click on the appropriate button on the **Arrange** toolbar for align position.

You can make multiple objects in your display a uniform size, width, and height. This function applies to drawing objects, such as boxes, circles, and ellipses, push buttons, process points, time/date objects, text objects, sliders, imported graphics, and TrendWorX embedded windows.

On the **Arrange** menu, the **Make Same Size** function enables you to make all selected objects the same size (with respect to width, height, or both) as the object of comparison.

To make objects the same size:

1. Select the objects you wish to make a uniform size.
2. Each object that you select is surrounded by a box with white handles. Left-click the focus object. The selection box around the focus object now has blue handles.
3. Select **Make Same Size** from the **Arrange**, menu or click the **Make Same Size** button on the **Arrange** toolbar. All objects size to the focus object immediately.

Make Same Height	
Make Same Width	
Make Same Size	

Bring to Front



Select **Bring to Front** on the **Arrange** menu, or click the **Bring to Front** button on the **Arrange** toolbar, to move the selected objects on top of the objects associated with it in the current work area.

Send to Back



Select **Send to Back** on the **Arrange** menu, or click the **Send to Back** button on the **Arrange** toolbar, to move the selected objects behind all the objects in the current work area.

Bring Forward



Select **Bring Forward** on the **Arrange** menu, or click the **Bring Forward** button on the **Arrange** toolbar, to move the selected objects up one layer in the display.

Send Backward



Select **Move Backward** on the **Arrange** menu, or click the **Move Backward** button on the **Arrange** toolbar, to move the selected objects down one layer in the display.

Rotation

The **Rotation** function allows you to rotate a selected object freely, rotate left and right, and flip the object horizontally and vertically.

Free Rotation of Static Objects



Free rotation of static objects means that the objects can be rotated to any angle. The following objects can be freely rotated

- Polylines
- Rectangles
- Ellipses
- Arcs
- Symbols
- Images

Objects can be freely rotated through the **Angle** field of Property Inspector or through the **Free Rotate** button on the **Arrange** toolbar.

Note: Objects that have been freely rotated can be flipped using the two flip buttons (see the **Horizontal/Vertical** Section).

Rotate Left and Right



Clicking the **Rotate Left** and **Rotate Right** buttons on the **Arrange** toolbar, shown above, allows you to rotate the selected object(s) to the left or the right in 90-degree increments. The following objects can be rotated by 90 degrees:

- Polylines
- Rectangles
- Ellipses
- Arcs
- Symbols
- Images
- Text

The ability to rotate text applies to the following:

- Multiline text
- Left, center, and right alignments
- Text objects
- Process points
- Data entries
- Statefields
- Timedates
- Buttons

Check boxes and radio buttons cannot be rotated.

Flip Horizontal/Vertical



Clicking on the **Flip Horizontal** and **Flip Vertical** buttons on the **Arrange** toolbar flips the selected object(s) to its horizontal or vertical axis. The following objects can be flipped:

- Polylines
- Rectangles
- Ellipses
- Arcs
- Symbols
- Images

Text and metafiles cannot be flipped.

Section 5

Display Edit Functions

Introduction to Editing Functions

This section describes the GraphWorX **Edit** menu commands for altering your display objects, including the following:

- Undo
- Cut
- Copy
- Paste
- Delete
- Paste Special
- Duplicate
- Select all objects
- Select all of type
- Select all dynamics
- Delete dynamics
- Find
- Replace
- Update shared objects
- Insert new objects

Undo Set Fill Color	Ctrl+Z
Can't Redo	
Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
Delete	Del
Paste Special...	
Duplicate	Ctrl+D
Select All Objects	Ctrl+A
Select all of type: Rectangle	
Select All Dynamics	
Delete Selected Dynamics	
Delete All Dynamics	
Find...	Ctrl+F
Replace...	Ctrl+E
Report...	
Update Shared Objects...	
Insert New Object...	
Links...	
Object	
Property Inspector...	

Figure 5.1. Edit Menu

Undo/Redo



Select **Undo** from the **Edit** menu to reverse the last editing action. Select **Redo** from the **Edit** menu to restore the last editing action that was undone. The name of the command changes depending on what the last action was. Select **Undo/Redo** from the **Edit** menu, or click the appropriate button on the **Main** toolbar. The **Undo** command changes to **Can't Undo** on the menu if you cannot reverse your last action.

Note: To undo several editing actions, continue to click the **Undo** button until all the edits you wish to remove have been undone. The same applies to the **Redo** command.

Shortcuts:



Keys: **CTRL+Z** or **CTRL+Y**

Basic Editing

Basic editing functions available on the **Edit** menu include **Cut**, **Copy**, **Paste**, **Paste Special**, and **Duplicate**.

Copy



Select **Copy** from the **Edit** menu to copy selected data/object to the clipboard. This command is unavailable if no data/objects are currently selected.

To copy an object:

1. Select the object. A box with handles surrounds the object.
2. Select **Copy** from the **Edit** menu, or click the **Copy** button on the **Main** toolbar.

Copying data to the Clipboard replaces the contents previously stored there.

Shortcuts:

Toolbar:



Keys:

CTRL+C

Cut



Select **Cut** from the **Edit** menu to move one or more selected objects to the windows Clipboard, removing them from the current display. Cutting data/objects to the clipboard replaces the contents previously stored there.

To cut an object:

1. Select the object to cut. A box with square handles surrounds the object.
2. Select **Cut** from the **Edit** menu, or click the **Cut** button on the **Main** toolbar. The object is removed from the display and sent to the Clipboard (here it remains until another object is placed onto the Clipboard.).

Note: You can also use the **DEL** key to cut selected objects and place them onto the Clipboard.

Shortcuts:

Toolbar:



Keys:

CTRL+X

Paste



Select **Paste** from the **Edit** menu to place a copy of the objects currently in the Clipboard into the work area. The objects remain on the Clipboard until you copy or cut another object to the Clipboard. Use this command to insert a copy of the Clipboard contents at the insertion point. This command is unavailable if the Clipboard is empty.

To paste an object from the clipboard:

1. Select **Paste** from the **Edit** menu, or click the **Paste** button on the **Main** toolbar.
2. The object is pasted near the center of the GraphWorX display work area and is enclosed in a select box. Use the mouse pointer to move the object to another place in the work area if desired.

Shortcuts:

Toolbar:



Keys:

SHIFT+INS

Paste Special

Select **Paste Special** from the **Edit** menu to paste objects in a specified format when more than one format is available. For example, some applications may copy images as both images and metafiles. The **Paste Special** dialog box, shown below, allows you to choose whether to paste the object as an image or metafile or as a device independent image.

If the **Display As Icon** box is checked, the image is displayed as an icon on your desktop.

You can paste the object if **Paste** is selected, or you can paste the link to the object if **Paste Link** is selected.

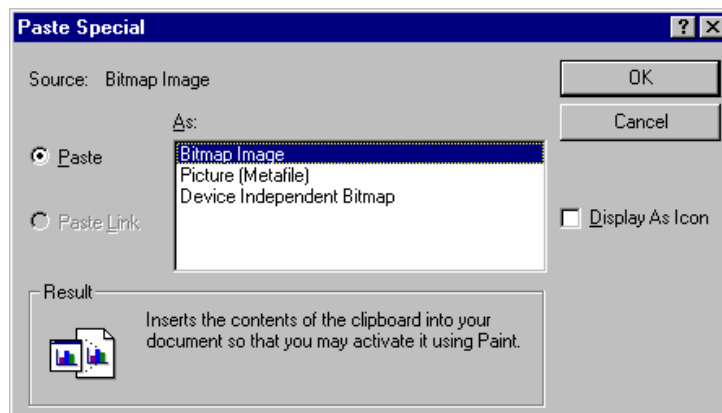


Figure 5.2. Paste Special Dialog Box

Duplicate



Select **Duplicate** from the **Edit** menu to duplicate a selected object.

To duplicate an object:

1. Select the object. A box with square handles surrounds the object.
2. Select **Duplicate** from the **Edit** menu, or click the **Duplicate** button on the **Main** toolbar. When an object is duplicated, all of its attending properties accompany the duplicated object.

Shortcuts:

Toolbar:



Keys:

CTRL+D

Select All

Select All Objects

Choosing **Select All Objects** from the **Edit** menu highlights all objects in the display.

Select All of Type

Choosing **Select All of Type** from the **Edit** menu allows you to select all objects of a specific type contained inside the display. The type selected depends on the tool currently selected on the toolbar and can be one of the following:

- Line
- Rectangle
- Ellipse
- Arc
- Poly-line
- Text
- Picture
- Metafile
- Process Point
- Time/Date
- Push Button
- Check button
- Radio Button

For example, if the **Rectangle** button is currently selected on the **Draw** toolbar, then the **Select All of Type** item on the **Edit** menu appears as shown in the figure below.

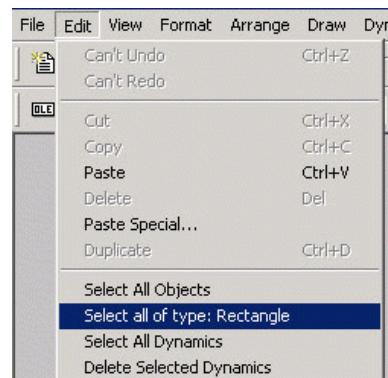


Figure 5.3. 'Select All Of Type' Command on Edit Menu

Select All Dynamics

Choosing **Select All Objects** from the **Edit** menu highlights all dynamic objects in the display (e.g. process points, time/date, etc.).

Find, Report, and Replace

The **Find** and **Report** functions on the **Edit** menu allow you to find and report strings in GraphWorX. Selecting **Find** from the **Edit** menu opens the **Find** dialog box as shown in the figure below. Enter a string to search for in the **Find What** field.

If **Match Case** is checked, this function finds/replaces only text strings that match the case of the characters. Otherwise the command finds/replaces strings with either uppercase or lowercase character that match the character in the **Find What** string.

If **Match whole word only** is checked, this function finds/replaces the entire text strings. Otherwise the command find/replace selects any string that contains a **Find What** substring.

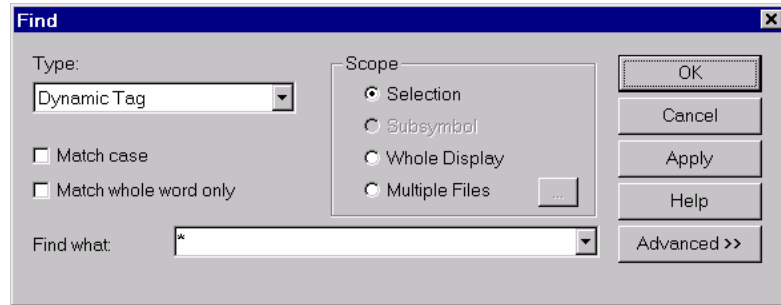


Figure 5.4. Find Dialog Box

Choose one of the following object types to search from the drop-down list under **Type**:

- Dynamic Tag
- Text Label
- Object Name
- Share Keyword
- Description
- Custom Data
- File Name

Under **Scope**, you can limit the search to one of the following:

- **Selection**: Searches the object that is currently selected in the display.
- **Subsymbol**: Searches a subsymbol in the display.
- **Current Layer**: Searches the current layer (in displays with multiple layers).
- **Whole Display**: Searches all items throughout the entire display.
- **Multiple Files**: Searches a group of display files. To specify the files to search, click the ... button next to the **Multiple Files** radio button. This opens the **Select Files** dialog box, as shown in the figure below. Click the **Add** button to choose a file. All selected files are shown under the **File List**.

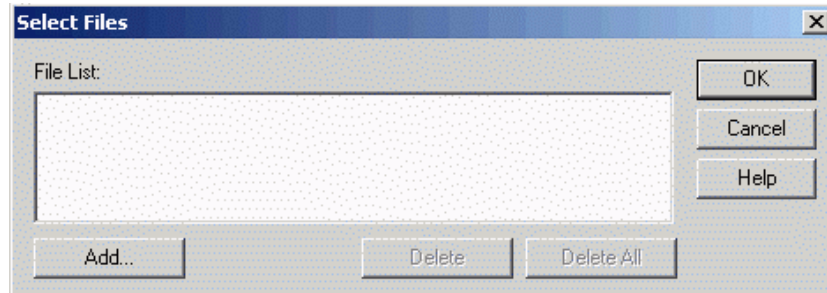


Figure 5.5. Selecting Multiple Files to Search

Click the **Apply** button on the **Find** dialog to perform the search. To view the search results, click the **Advanced** button. This expands the **Find** dialog box to show the search results, as shown in the figure below. The results are displayed as a tree view. Selecting an item in the results tree and clicking the **Properties** button opens the Property Inspector for the selected item. Click the **Show Selection** button to view the item in the display. In the Multiple File scope, selecting an object not in the current display loads a new display with the object selected.

Note: If the type is **Object Name** and the **Find What** string is ***.***, the results tree view displays all object hierarchy from current Scope (including the Multiple Files scope).

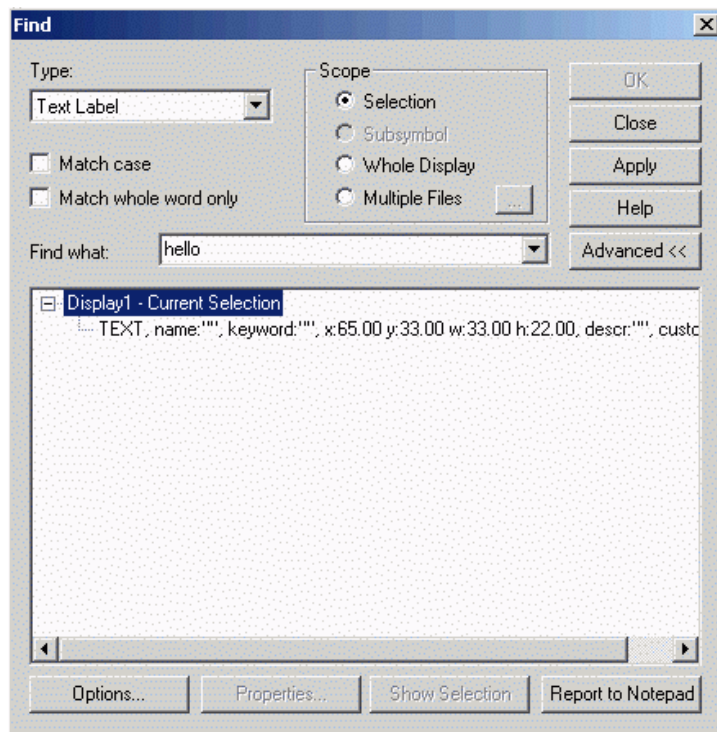


Figure 5.6. Find Dialog Box: Results View

You can customize the results view by clicking the **Options** button. This opens the **Report Options** dialog box, as shown in the figure below. Choose the items to include in the results tree. You can also choose to identify items in the results by enclosing the items in double or single quotes.

To generate a report for the Find operation, click the **Report to Notepad** button on the **Find** or **Report** dialog boxes. This exports the search results to a text file.

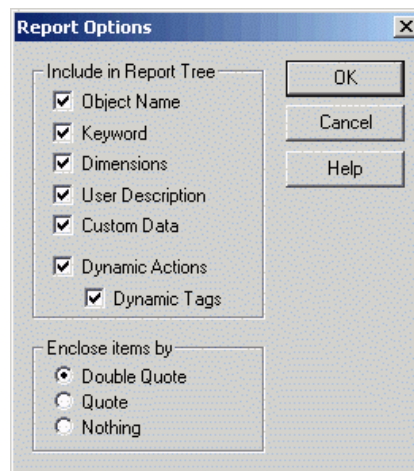


Figure 5.7. Report Options Dialog Box

The **Replace** operation works in a similar way and replaces the **Find What** string(s) with the **Replace With** string(s). All dialog options are the same, except that the **Replace** dialog box has a field called **Replace With**, which also allows you to select from the **Tags Menu** when **Dynamic Tag** is selected under **Type**.

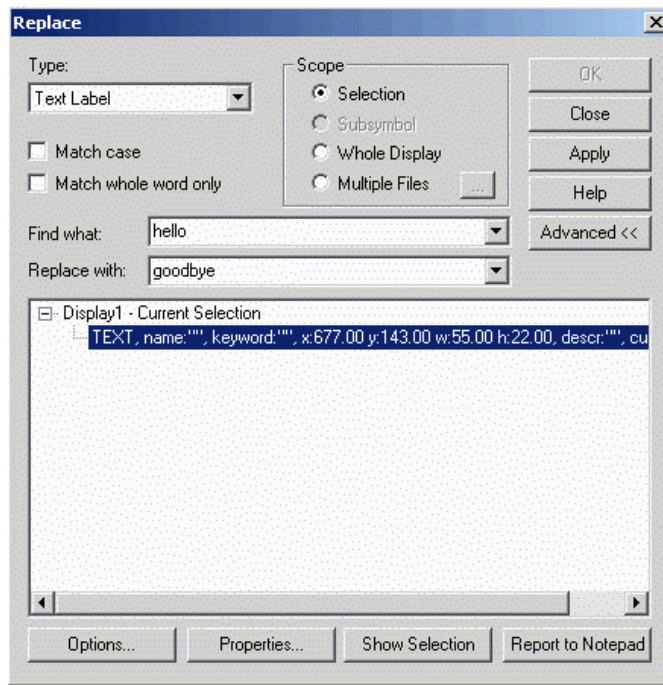


Figure 5.8. Replace Dialog Box

User Interface for Find and Replace

Find/Replace User Interface

Parameter	Description
Type	Specifies a type of string where the Find and Replace commands work. All the strings can be modified for every object or dynamic action through the Property Inspector dialog box.
Dynamic Tag	Finds/replaces tags (dynamics points).
Text Label	Finds/replaces labels of text objects.
Object Name	Finds/replaces names of objects/dynamic actions.
Share Keyword	Finds/replaces global keywords used in shared objects.
Description	Finds/replaces based on a description.
Custom Data	Finds/replaces custom data.
File Name	Finds/replaces file names.
Scope	Specifies where to find/replace items.
Selection	Searches in the currently selected item in the display. If no object is selected, this option is disabled.
Subsymbol	Searches a subsymbol in the display.
Whole Display	Searches all items throughout the entire display.
Multiple Files	Searches a group of display files. To specify the files to search, click the ... button next to the Multiple Files radio button.
Match Case	If checked, this function finds/replaces only text strings that match the case of the characters. Otherwise the command finds/replaces strings with either uppercase or lowercase character that match the character in the "Find What" string.
Match Whole Word Only	If checked, this function finds/replaces the entire text strings. Otherwise the command find/replace selects any string that contains a "Find What" substring.
Find what	Specifies the search text. A drop-down list allows you to select from all strings listed. This set of strings is determined by the Scope and Type fields. Wildcards are allowed (see Wildcards below). Note that if the type is Object Name and the Find What string is '*', the results tree view displays all object hierarchy from current Scope (including multiple files scope).

Parameter	Description
Replace with	Specifies the string to replace the characters found. A drop-down list allows you to select from all possible strings. This set of strings is determined by the Scope and by the Type radios. Wildcards do not work here (they have no special meaning).
Results View	Tree view shows the result of the find/replace operation. All the object hierarchy that matches the Scope and "Find What" string is displayed as a tree control. If a leaf (an object or symbol) is selected, clicking the Show Selection button selects the specified object in the display. Note that in the multiple file scope, selecting an object not in the current display loads a new display with the object selected.
OK	Starts the find/replace operation. The matching objects are selected in the display and the dialog is closed.
Cancel/Close	Cancels the dialog and closes it. If the Apply button is clicked, the label is changed from Cancel to Close, and the state of the dialog from the last Apply is saved.
Apply	Starts the find/replace operation. The dialog is not closed so the result can be inspected in the results tree view.
Advanced	Toggles the results tree view and enables the Options, Properties, Show Selection, and Report to Notepad buttons.
Options...	Opens the Report Options dialog box, which customizes the results view settings.
Properties	Opens the Property Inspector for the selected item in the results view.
Show Selection	Displays the selected item in the results view.
Report to Notepad	Exports the search results to a text file.

Wildcards

"Wildcard characters" '*' and '?' can be used in a "Find What" string. Character '*' can be used to match a group of characters. Character '?' can be used instead of one character. Display all objects from current scope by choosing the Type - Object Name and entering an asterisk (*) to the "Find What" combo.

Insert Object

Selecting **Insert New Object** from the **Edit** menu opens the **Insert Object** dialog box, shown below. This function inserts and embeds an OLE object, such as a chart or an equation, into the current display. The application in which the object was created becomes active on the screen. The **Insert Object** command also allows you to insert an ActiveX control into the current display. The property sheet for that ActiveX control is automatically opened after inserting the ActiveX control.

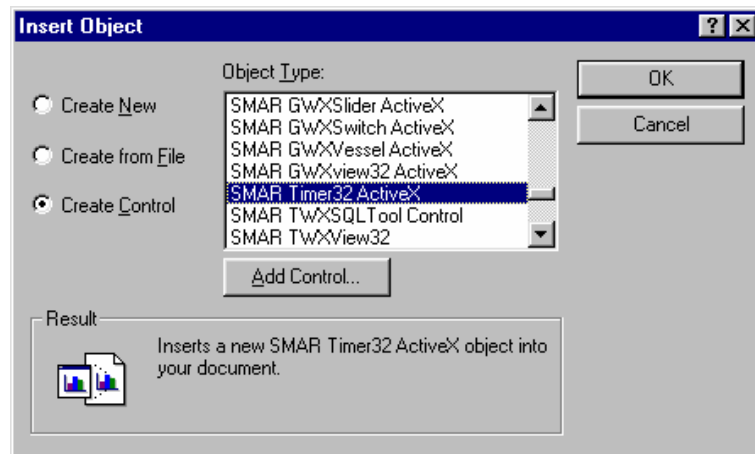


Figure 5.9. Insert Object Dialog Box

To insert a new object into the display:

1. Choose either **Create New** or **Create From File**. **Create New** inserts a new OLE object from the list provided. Some of these objects have default values, which can be changed using a property editor. When **Create From File** is selected, you can enter the file name in the field provided, or you can click the **Browse** button to search for the file. When **Create Control** is selected, you can click the **Add Control** button to browse for an object.
2. Select the object type.
3. Click **OK**.

GraphWorX checks the version of ActiveX controls embedded in the display and installed on the computer and reports any potential conflicts, as shown in the figure below.

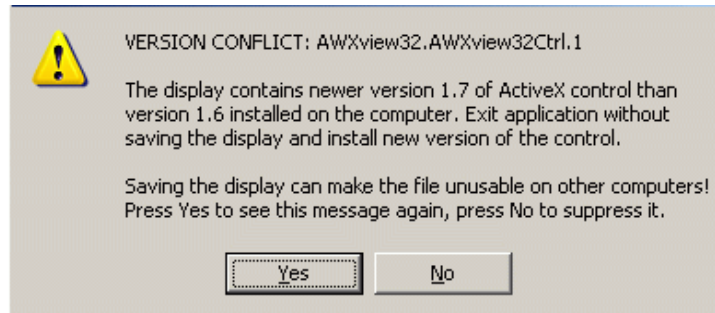


Figure 5.10. Version Conflict With Embedded ActiveX Control

Once an object (e.g. TrendWorX Viewer ActiveX) is inserted into the display, the Object properties are enabled on the **Edit** menu, as shown in the figure below. Selecting **Object > Properties** from the **Edit** menu opens the Properties dialog box for the object.

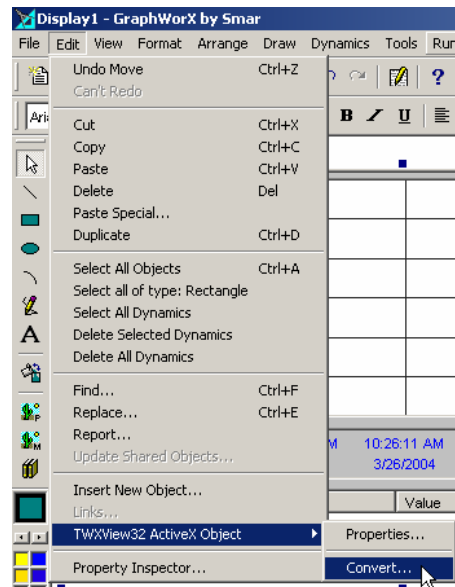


Figure 5.11. Object Commands on Edit Menu

Selecting **Object > Convert** from the **Edit** menu opens the **Convert** dialog box, as shown in the figure below. To convert an object, select the **Object Type** and choose **Activate as**.

If an object is large and you want to reduce the amount of space that the object takes up in the display, check the **Display As Icon** check box. This replaces the object in the display with an icon of your choice. To choose an icon different from the standard icon, click the **Change Icon** button.

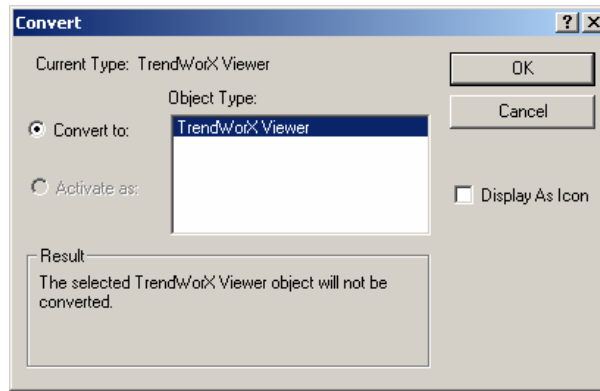


Figure 5.12. Converting an Object

Update Shared Objects

The **Update Shared Objects** function is accessible from the **Edit** menu. Selections made in the Property Inspector determine shared objects. An object can be shared by using the **Shared Keyword** feature in the Property Inspector. It is made to belong to the desired group by entering a keyword shared by all the objects of one group. Any change can be made to an object of the group and can be applied to the rest of objects in the specified group using **Update Shared Objects**.

A least one object from a group must be changed to enable the **Update Shared Objects** command. Thus, if several objects are integrated into a symbol, that symbol is duplicated. If the properties of one object in the symbol group are changed, the changes are applied globally to the duplicated object by using the **Update Shared Objects** function, once a share keyword has been specified.

Selecting **Update Shared Objects** from the **Edit** menu opens the **Update Shared Objects** dialog box, as shown in the figure below.

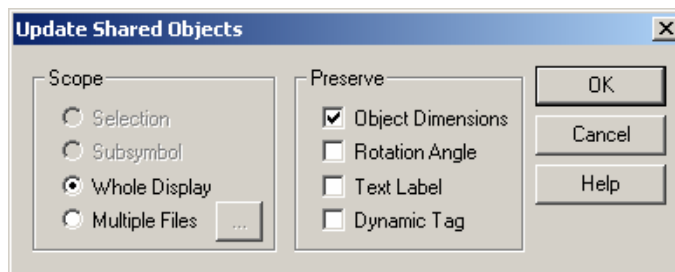


Figure 5.13. Update Shared Objects Dialog Box

User Interface for Update Shared Objects

Update Shared Objects Dialog Box Parameters

Parameter	Description
Scope	Determines the operating space for the operation.
Selection	Updates the currently selected item in the display.
Subsymbol	Updates a subsymbol (see menu item Edit Symbol and Subsymbol Editing capability); if there is no subsymbol edited, this option is disabled.
Whole Display	Updates the entire display; any subsymbol editing is left and replaced by a root edit level.
Multiple Files	Updates a group of file display files. To specify the files to search, click the ... button next to the Multiple Files radio button. Any subsymbol editing is left and replaced by root level. If there are changes in the current document, a dialog appears asking to save them or cancel the operation.
Preserve	If you do not want to update all aspects of an object, you can preserve (not update) any of the following by checking the appropriate check boxes: Object Dimensions, Rotation angles, Text Labels and Dynamic Tags.

Where and How to Use Shared Objects

Consider a display with many inputs and outputs. If all input label objects are marked as shared with, for example, INPUT keyword, and all output label objects are marked as shared with, for example, OUTPUT keyword, it is very fast and easy to modify the color, size, 3D effect, and shadow for all of them. Just do the change on one object and apply it to the rest. (Remember to check the **Preserve Text Labels** check box.)

Consider a project with many pumps. If the displays in the project are created carefully from the beginning, and all desired pumps are marked as shared with same keyword, it is fast and easy to change the appearance. If the pumps are connected to different signals, remember to check the **Preserve Text Labels** check box on the **Update Shared Objects** dialog box. For changes to tags, consider using the Find/Replace capability with wildcards to select the desired group of signals to be searched or replaced.

Consider a company logo or some similar pictures placed in many displays in different positions (if placed in the same position, templates should be the right tool for that job). By one operation, all of them can be modified or increased in size. To make only one larger, use the **Update Share Objects** function, and remember to uncheck **Preserve Object Dimensions** on the **Update Shared Objects** dialog box.

Note:

Viewing Files

View Functions

The following sections describe each View menu function shown below. Many functions can also be accessed through the View toolbar.



Figure 6.1. View Menu



Figure 6.2. View Toolbar

The Home View



The **Home** view is defined as the view in which you last saved the display.

To restore the viewing area to the home view, select **Home** from the **View** menu. The display work area that in which the display was last saved is restored to the screen. The zoom factor appears on the bottom right of the screen.

Zoom Functions

The **Zoom** factor is indicated in the bottom right portion of the status bar. 100% zoom, 1 pixel = 1 unit is the default.

Zoom

To zoom a display, select **Zoom** from the **View** menu, or select one of the buttons shown below from the **View** toolbar. The **Zoom** submenu on the **View** menu allows you to choose from a default zoom value, to zoom in or out, or to type in a custom zoom percentage.



Zooms in by increments of 50 percent.

Zooms out by increments of 50 percent.

Allows you to enter a custom percentage to zoom.

Custom Zoom

To set a custom zoom percentage, select **Zoom > Custom** from the **View** menu. This opens the Custom Zoom dialog box. Enter a percentage in the **New Zoom** field, and then click **OK**.

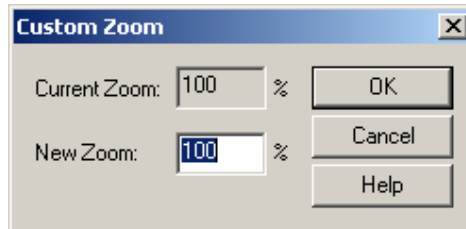


Figure 6.3. Setting a Custom Zoom Percentage

Unzoom



The unzoom function acts as an undo for zoom. Select **Unzoom** from the **View** menu, or click the **Unzoom** button on the **View** toolbar.

Box Zoom



The **Box Zoom** command allows you to select an area on which to zoom.

To use the box zoom function:

1. Select **Box Zoom** from the **View** menu. The cursor appears as a magnifying glass, indicating the zoom option is enabled.
2. Click the left mouse button and drag to draw a box around the area in the display selected to zoom, and then release the mouse button. Or, click the left mouse button without dragging to zoom in (plus 50 percent) on the point on which you clicked. Click the right mouse button without dragging to zoom out (minus 50 percent) on the point clicked on.
3. The area appears in zoomed view. You can zoom in numerous times.
4. Select **Home** from the **View** menu when you are done viewing that display area in the zoom mode and want to return to the home view. Or select **Unzoom** from the **View** menu to undo your zoomed views.

Setting Zoom Limits for Box Zoom

The Box Zoom feature allows you to select an area to which the current view should zoom. However, it also allows you to zoom in by left-clicking, or to zoom out by right-clicking. You can specify limits for the Box Zoom, so the operator using the Box Zoom does not run out of preferred zoom, using the following registry entries:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\ICONICS\GWX32\Runtime Settings]
BoxZoomMaxPercent:    DWORD .. maximal allowed zoom, by default 999 (%)
BoxZoomMinPercent:    DWORD .. minimal allowed zoom, by default 1 (%)
```

Example:

To configure the Box Zoom to zoom between 30% and 150%, set **BoxZoomMinPercent** to 30 and **BoxZoomMaxPercent** to 150%.

Zoom Selection



Click the **Zoom Selection** from the **View** menu, or click the **Zoom Selection** button on the **View** toolbar, to zoom in on the selected object(s).

Fit to Window



Select **Fit to Window** from the **View** menu, or click the **Fit to Window** button on the **View** toolbar, to size the display such that all objects in the display fit inside the window.

Show Whole Display



Select **Show Whole Display** from the **View** menu, or click the **Show Whole Display** button on the **View** toolbar to see the entire work area.

Decluttering Zoom

The **Decluttering Zoom** option on the **View** menu allows layers to be hidden when they are zoomed in or zoomed out beyond their zoom threshold. For this feature to work, you must set the zoom thresholds by selecting **Layers > Edit Layer Properties** from the **Format** menu and specifying the **Zoom Visibility Range** in the **Edit Layer Properties** dialog box.

Mouse Zoom

A mouse zoom feature has been added to GraphWorX to simplify the display navigation:

- To zoom in, press the **ALT** key and roll the mouse wheel forward.
- To zoom out, press the **ALT** key and roll the mouse wheel backward.
- To pan the screen press the **ALT** key, push the left mouse button, and drag the mouse.

The zoom is selection-sensitive so that if an object is selected, the display automatically recenters the screen to the object. This also applies to multi-selection of display objects.

Summary Information

Selecting **Summary Information** from the **View** menu opens the **Summary Information** dialog box, shown below.

The image shows a screenshot of the 'Summary Information' dialog box. The dialog has a title bar with 'Summary Information' and a close button. It contains two tabs: 'Summary' and 'Statistics'. The 'Summary' tab is selected. The dialog contains the following fields and text:

- Application: GraphWorX
- Author: Bob Jones
- Keywords: (empty text box)
- Comments: This display provides the interface to the recirculation system.
- Title: (empty text box)
- Subject: Recirculation System
- Template: (empty text box)

At the bottom of the dialog are four buttons: OK, Cancel, Apply, and Help.

Figure 6.4. Summary Information Dialog Box

Summary Tab

In the **Summary** tab, you can fill in the details such as **Author**, **Keywords**, **Comments**, **Title**, and **Subject** in the spaces provided.

Statistics Tab

The **Statistics** tab, shown below, provides general information about the file currently on display.

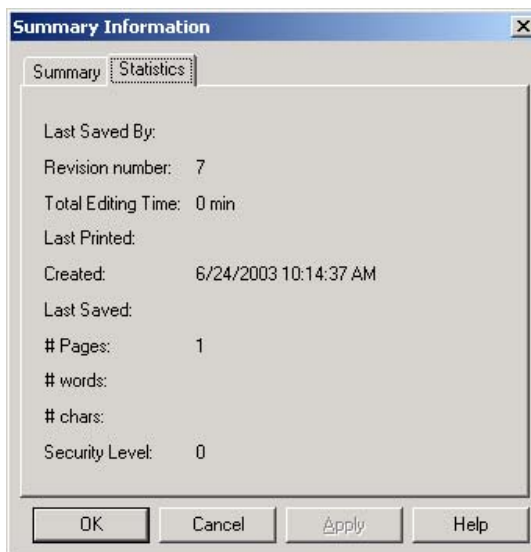


Figure 6.5. Statistics Tab

Note

Summary Information for a display can also be viewed in Windows Explorer. In Windows Explorer, right-click the display file and choose **Properties** from the pop-up menu. The **Summary** and **Statistics** tabs will appear in the dialog presented by Windows Explorer.

Viewing Object Count Statistics

The **Object Count** function on the **View** menu displays statistics for objects, actions, and display sizes in your displays.

To display object statistics:

1. Select **Object Count** from the **View** menu. This opens the **Object Count for Current Display** dialog box, shown below, which contains data about static and dynamic objects.
2. Click the **OK** to return to the display.

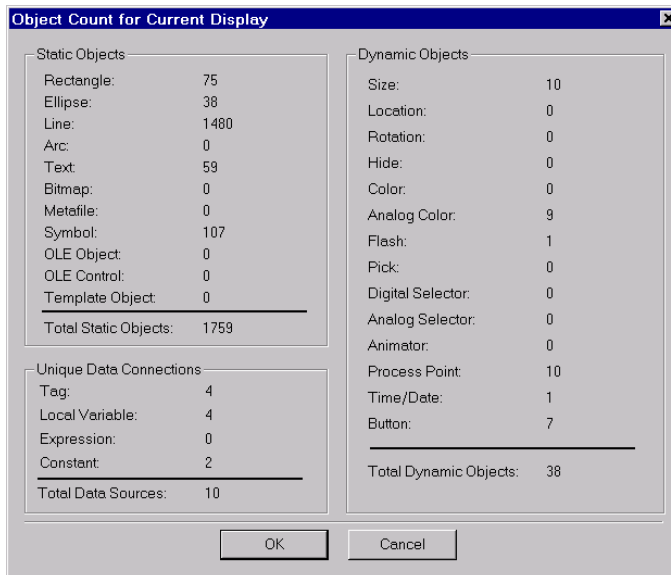


Figure 6.6. Viewing Object Count Statistics

Object Count Parameters

Parameter	Description
Static Objects	
Rectangle	Number of rectangles in the display.
Ellipse	Number of ellipses in the display.
Line	Number of lines in the display.
Arc	Number of arcs in the display.
Text	Number of text objects in the display.
Bitmap	Number of images in the display.
Metafile	Number of metafiles used in the display.
Symbol	Number of symbols used in the display.
OLE Objects	Number of OLE Objects in the display.
OLE Control	Number of OLE Controls in the display.
Template Object	Number of Template Objects in the display.
Total Static Objects	Total number of static objects in the display.

Unique Data Connections	
Tags	Number of Tags in the display.
Local Variable	Number of Local Variables in the display.
Expression	Number of Expressions in the display.
Constant	Number of Constants in the display.
Total Data Sources	Total number of data sources in the display.

Dynamic Objects	
Size	Number of Size connections in the display.
Location	Number of Location connections in the display.
Rotation	Number of Rotation connections in the display.
Hide	Number of Hide connections in the display.
Color	Number of Color connections in the display.
Analog Color	Number of Analog Color connections in the display.
Flash	Number of Flash connections in the display.
Pick	Number of Pick connections in the display.
Digital Selector	Number of Digital Selector connections in the display.
Analog Selector	Number of Analog Selector connections in the display.
Animator	Number of Animator Actions connections in the display.
Process pt.	Number of Process Points in the display.
Time/Date	Number of Time and Date fields in the display.
Button	Number of pushbuttons in the display.
Total Dynamic Objects	Total number of dynamic objects in the display.

Hide Layers

You can hide or show layers by selecting **Hide Layers** from the **View** menu or the keyboard shortcut combination **CTRL+SHIFT+0**. Both of these options open the **Check Layers to Hide** dialog box, shown below. Select a layer or layers to hide, and then click **OK**.

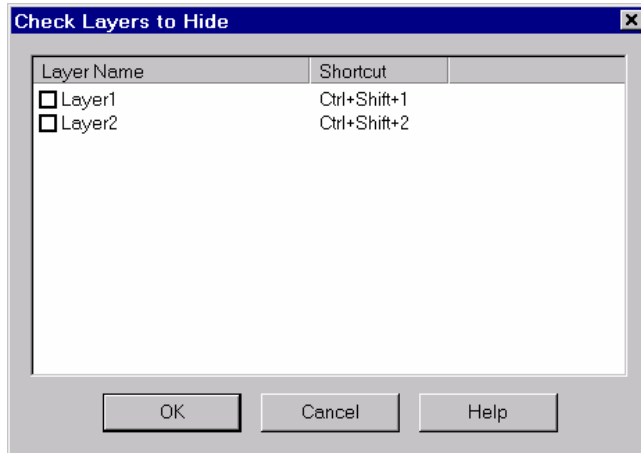


Figure 6.7. Selecting Which Layers to Hide

Layers may also be hidden due to the value of the tag to which they are attached, as well as if you select **Layers > Edit Layer Properties** from the **Format** menu and select the **Explicitly Hide Layer** check box in the **Edit Layer Properties** dialog box, as shown in the figure below. The **Explicitly Hide Layer** feature forces the layer to remain hidden regardless of whether the layer would be visible based on the zoom threshold. The layer will remain hidden until this option is turned off, at which point the layer may remain hidden depending on the attached tag value and zoom threshold.

Each layer that you added during configuration is shown in the dialog box, along with the system generated shortcut key. Note that there is a check box to the left of each layer name. If this box is checked, that layer will be hidden during runtime mode.

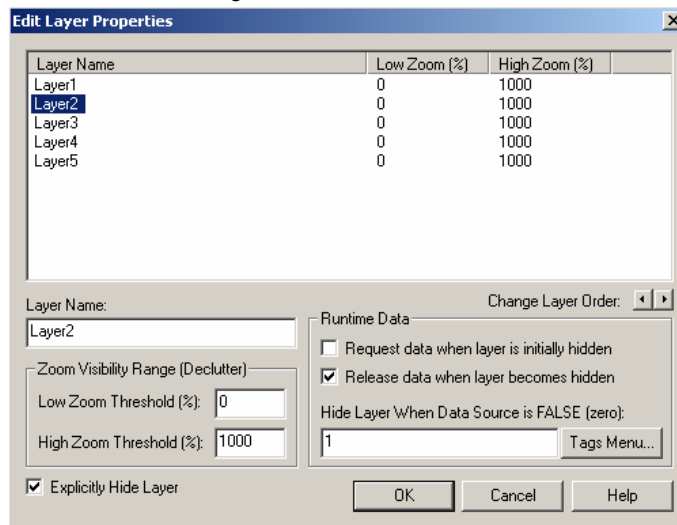


Figure 6.8. Explicitly Hide Layers Function in Layer Properties

Toolbars

The **Toolbars** command on the **View** menu opens the **Show Toolbars** dialog box, shown below, which allows you to select which toolbars you want to be visible. The **Toggle Toolbars** command on the **View** menu hides or shows all of the toolbars at once.

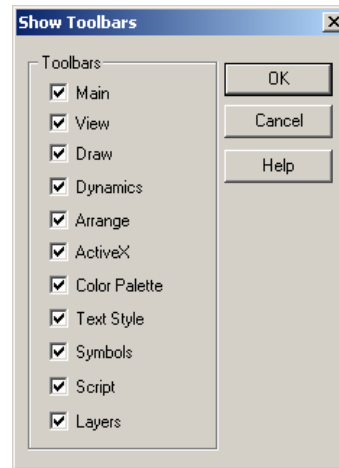


Figure 6.9. Show Toolbars Dialog

Viewing Current Coordinates on the Status Bar

The current mouse coordinates are shown in the right-hand corner of the status bar. The current coordinates identify the position of the mouse pointer in the work area through X- and Y-coordinates, as shown in the figure below. This is helpful if you are working in pixels and need to define the mouse pointer position more accurately.

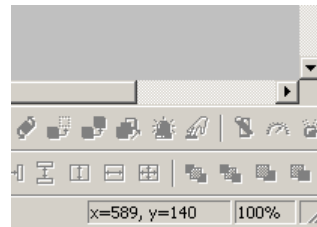


Figure 6.10. Viewing Mouse Coordinates on the Status Bar

Scrolling

The scrolling function allows you to scroll up and down your display using the scroll bar displayed on the right of your screen. It also allows you to scroll left and right using the horizontal scroll bar displayed at the bottom of your screen.

Horizontal Scroll Bar and Vertical Scroll Bar

The **Horizontal Scroll Bar** and **Vertical Scroll Bar** commands on the **View** menu set the visibility of the horizontal and vertical scroll bars, respectively. The **Toggle Scroll Bar** command on the **View** menu shows or hides both scroll bars at once.

Grid

The **Grid** command on the **View** menu shows or hides the grid. The **Grid** tab in the **Application Preferences** and **Display Properties** dialog boxes, shown below, gives you flexibility with line style, color, and spacing.

The **Grid Size** is in arbitrary base units (depending upon the current zoom factor). When zoom is at 100%, the spacing is in pixels. The **Grid Size** specifies the grid spacing in terms of width and height.

The **Grid Style** allows you to choose a style of dots or dashes (lines). You can choose a **Vertical Line Style** or a **Horizontal Line Style** from the options available.

You can choose a **Grid Color** for the lines and dots of the grid. By checking the appropriate boxes, you can choose to **Show Grid** and **Snap to Grid**.

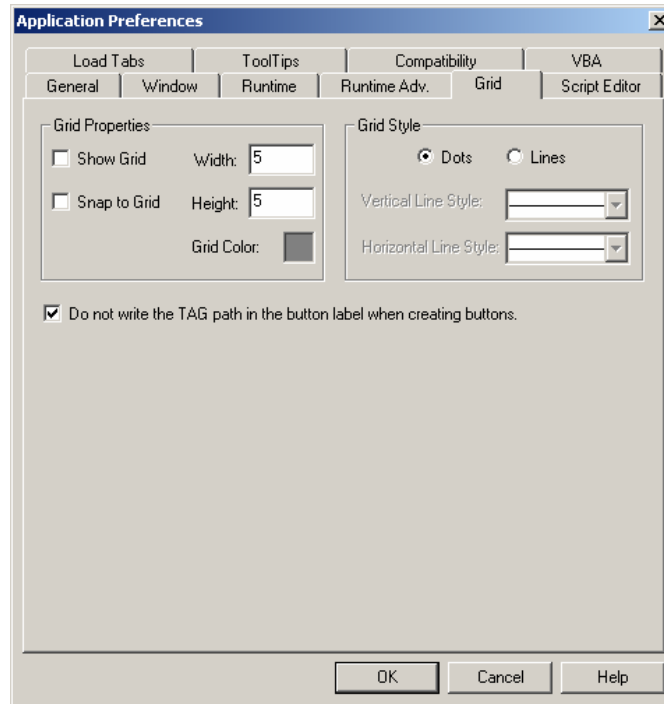


Figure 6.11. Application Preferences: Grid Tab

Runtime Window Properties Mode

Selecting **Runtime Window Properties Mode** from the **View** menu makes it possible to see what a display will look like in runtime mode without actually entering runtime. This helps you set the proper display dimensions and zoom factors. When the action toolbars are hidden, the configuration functions can be found in the **Main** menu or in the right-click popup menu.

Note: Pressing **CTRL+M** toggles the display between configuration and runtime mode.

Properties Window

Selecting **Properties Window** from the **View** menu opens the **Properties** list for the display, as shown below. This lists the property settings for the selected form or control. A **property** is a characteristic of an object, such as size, caption, or color.

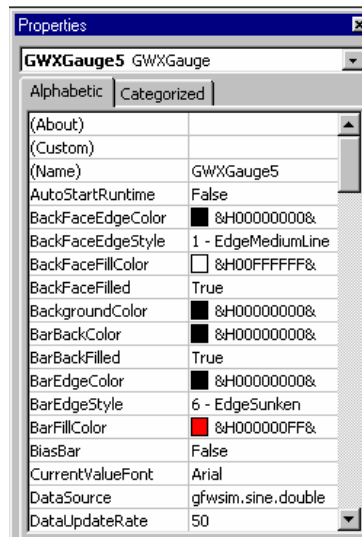


Figure 6.12. Properties Window

Select Language

The **Select Language** function on the **View** menu allows you to choose which language you want to use in your display. Choosing **Select Language** from the **View** menu opens the **Select Language** dialog box, shown below.



Figure 6.13. Select Language Dialog Box

Define the parameters listed in the table below. Then click **OK** to return to the work area.

Select Language Parameters

Parameter	Description
List	Lists available languages. Depending on which item you have selected, the view on the left will change. If English is checked, the languages will appear as their English name. If Localized is checked, the languages will appear with the native country in parentheses (for languages with several dialects only). When Native is checked, the languages are displayed the way they would be written in that language.
Installed Locales Only	If this is checked, local languages appear in the box.
Available Language Translations Only	Checking this box allows you to choose from available language translations only.

Data Connections

Introduction to Data Connections

This section discusses the interface of GraphWorX objects to the field Input/Output. The following topics will be discussed here:

Making data connections with the Tag Browser

Expressions

Constants and local variables

Creating generic reusable displays and symbols with aliasing

Dynamic animation is achieved by transforming visible objects based on specified data connections. Each dynamic object allows one or more data connections for its primary value(s). Some dynamics also allow range overrides; these range overrides are also data connections. A data connection is most likely to be a tag representing a value in the system. However, a data connection can also be a constant value, a local GraphWorX variable, or a mathematical expression (which can include tags, constants, and local variables).

Data connections can be typed directly into the data source edit control in the configuration dialogs of the various dynamics. Text strings can also be dragged and dropped into this edit control from any drag source, which makes the drag-and-drop data available in the Windows text clipboard format. There are also two buttons:

OPC Tags button - Opens the **Tag Browser**.

Tags Menu button - Displays a menu with the following options: Expression Editor, Aliases, Local Variables, Simulation Variables, and Global Aliases.

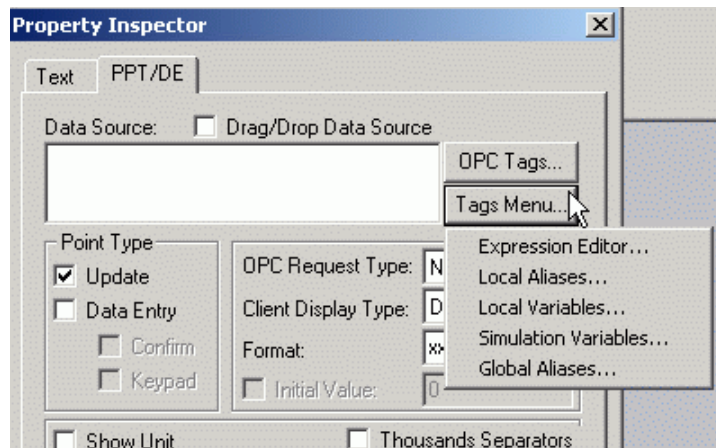


Figure 7.1. OPC Tags and Tags Menu Buttons in Property Inspector

Tags

All nonnumeric data-connection strings that do not include the special tokens described later in this section are interpreted as tags. If your tag happens to be a string that is a number (a rare situation) and you do not want the tag to be auto-detected as a constant value, use the following syntax:

```
{{tag_name}}
```

Any tags that are not defined in the system at runtime will point fail.

Tag Browser

Clicking the **Tags** button in the Property Inspector opens the **OPC Universal Tag Browser**, shown below. (See the Tag Browser documentation for more information.) This feature allows you to search for tags that specify certain available parameters.

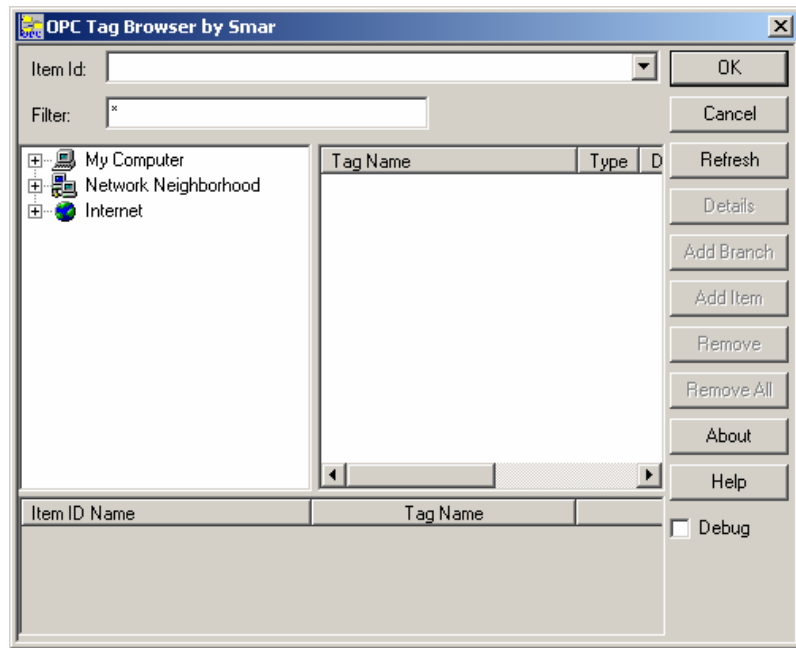


Figure 7.2. OPC Universal Tag Browser

Description of User Interface of the Tag Browser

Tag Browser Parameters

Parameter	Description
Item Id	Specifies the tag name and path.
Filter	Specifies wildcard characters (*) to select the desired tag.

Expression Editor

Clicking the **Tags Menu** button in the Property Inspector opens the menu shown below.

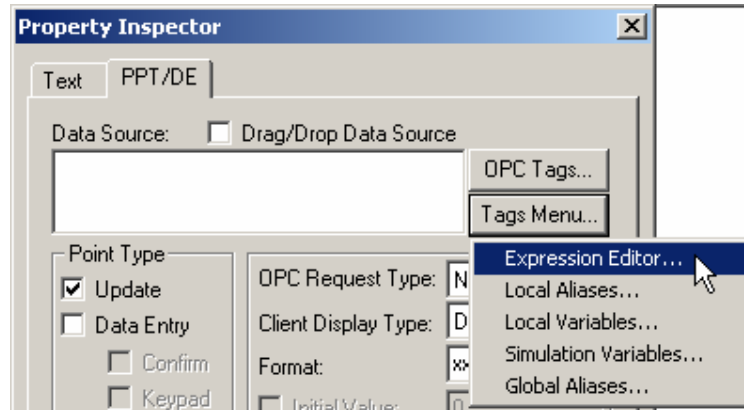


Figure 7.3. Tag Menu

Selecting **Expression Editor** from the **Tags** menu opens the **Edit Expression** dialog box, shown below. Categories of functionality available for expressions include:

- Arithmetic
- Relational
- Logical
- Bitwise
- Functions

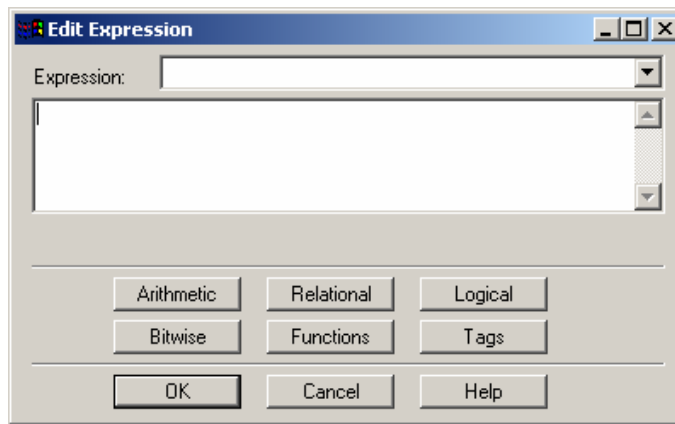


Figure 7.4. Edit Expression Dialog Box

Writing Expressions

The **Expression Editor**, shown below, is available to assist you in creating expressions for your ProcessView applications. The window is resizable and can be stretched as well as maximized or minimized. The drop-down list at the top of the **Edit Expression** dialog box keeps track of the last 50 expressions you have entered. The expression entered most recently is the first one in the drop-down list.

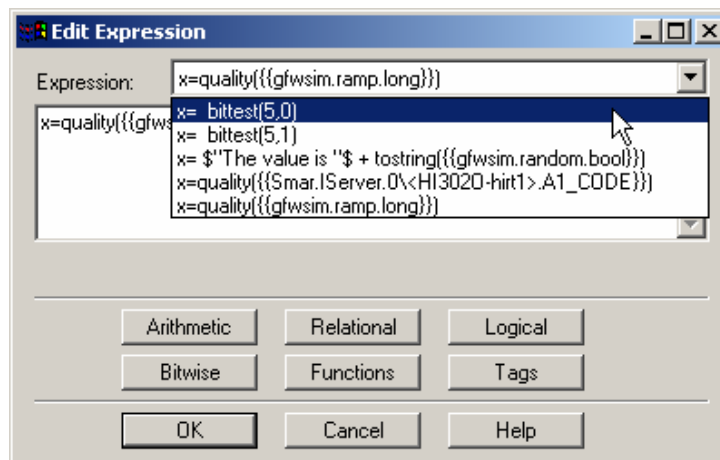


Figure 7.5. Expression Editor

An **expression** is a string that defines and evaluates a data connection between a client and an OPC server. During runtime mode, OPC servers resolve the data value for the expression. To indicate that a data connection is an expression, precede the string with the "x=" token, as shown below:

```
x={{Smar.Simulator.1\SimulatePLC.PumpSpeed}}
```

You can either type your expressions directly into the text box of the **Edit Expression** dialog box, or you can use the symbols and functions provided that help you use the proper string syntax when writing expressions. The following categories are available:

- Arithmetic
- Relational
- Logical
- Bitwise
- Functions
- Tags

Strings in Expressions

Expressions allow calculations to be performed on incoming data. The OPC server can provide the data in one or more data types, such as "float," "long," "integer," "string," etc. If the numeric data are coming from the server as strings, they are compared as strings in expressions. This is done based on the alphabetical order of the letters. Therefore, an expression evaluated as TRUE "20" > "100" correctly. Of course, if there were an expectation of a numeric comparison, $20 < 100$ and the above expressions might seem to be evaluated incorrectly, but they are not.

There is a workaround. If you add a numerical zero to each of the tags, the logic operators will work properly. For example:

```
x={{JC.N1OPC.1.0\HDQTRS\sys2\ad-3.Present Value}}+0) >
  ({{JC.N1OPC.1.0\HDQTRS\sys2\ad-4.Present Value}}+0)
```

An alternative way is to change the OPC server so that it sends the strings with a fixed number of digits with leading zeros, or to use DataWorX registers for a conversion from a string to a number.

Point Extension Syntax

The **Point Extension Syntax (PES)** allows for retrieving additional information related to OPC tags, such as quality and timestamp. The following are example expressions using a valid PES request:

```
tag:SMAR.Simulator\SimulatePLC.Ramp#timestamp
```

```
tag:SMAR.Simulator\SimulatePLC.Ramp#quality
```

```
tag:\pc1\SMAR.Simulator\SimulatePLC.Ramp#timestamp
```

```
tag:\pc1\SMAR.Simulator\SimulatePLC.Ramp#quality
```

Sometimes it may be necessary to enforce the "request data type" to a specific type, such as "string," in order to display this information in a process point.

OPC Tags

An **OPC tag**, or data point, is a data connection between a client and an OPC server. OPC tags can be used in expressions when the tag is embedded between double brackets, as shown below:

```
{{tag_name}}
```

Example:

```
x={{SMAR.Simulator.1\SimulatePLC.PumpSpeed}}
```

You can use the Tag Browser, shown below, to select OPC Alarm and Event (AE), Data Access (DA), and Historical Data Access (HDA) tags to include in your expressions.

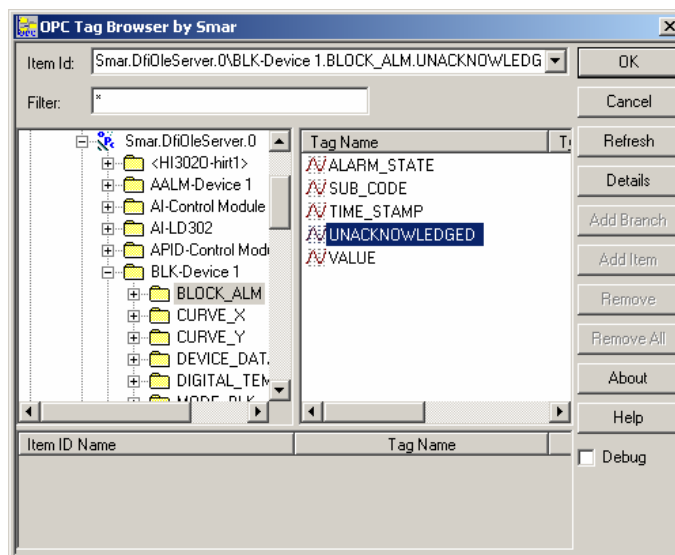


Figure 7.6. Tag Browser

Aliases

An **alias** is a string that represents or describes an object or data point in a display. Both local and global aliases can be used in expressions.

Local Aliases

For local aliases within the expression, use the following syntax:

```
<<local_alias_name>>
```

Example:

```
x=<<TankLevel>>
```

Global Aliases

For global aliases within the expression, use the following syntax:

```
<#global_alias_name#>
```

Example:

```
x=<#RoomTemperature#>
```

Selecting **Global Alias Browser** opens the Global Alias Browser, as shown in the figure below. Select a global alias from the Global Alias Browser, which includes all global aliases in the global alias database. This eliminates the need to manually type in the alias name. All global aliases that are configured in the Global Alias Engine Configurator are conveniently available to choose from inside the browser. The tree control of the Global Alias Engine Configurator is mimicked in the tree control of the Global Alias Browser. Select a global alias by double-clicking the alias name (e.g. "Floor" in the figure below). The alias name appears at the top of the browser, which automatically adds the <# and #> delimiters to the alias name. Click the **OK** button.

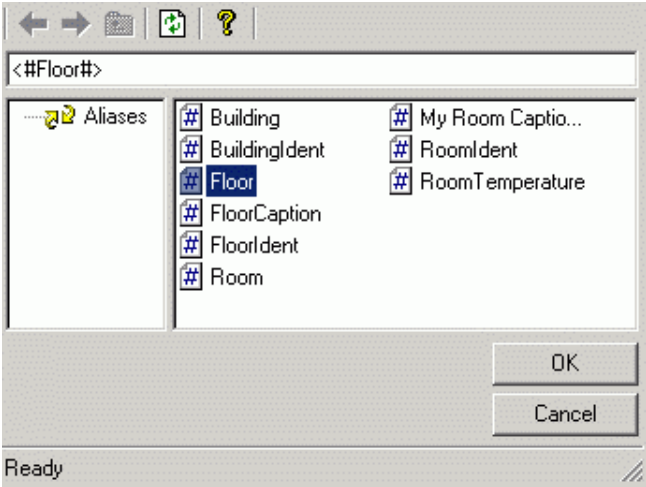


Figure 7.7. Selecting an Alias From the Global Alias Browser

Language Aliases

For language aliases within the expression, use the following syntax:

```
/+language_alias_name+/
```

Example:

```
x=/+WaterSystem+/
```

Selecting **Language Alias Browser** from the pop-up menu opens the Language Alias Browser, as shown in the figure below. The browser includes all languages aliases in the language database. All language aliases that are configured in the Language Configurator are conveniently available to choose from inside the browser. The tree control of the Language Configurator is mimicked in the tree control of the Language Alias Browser. Select a language alias by double-clicking the alias name. The alias name appears at the top of the browser, which automatically adds the /+ and +/ delimiters to the alias name. Click the **OK** button.

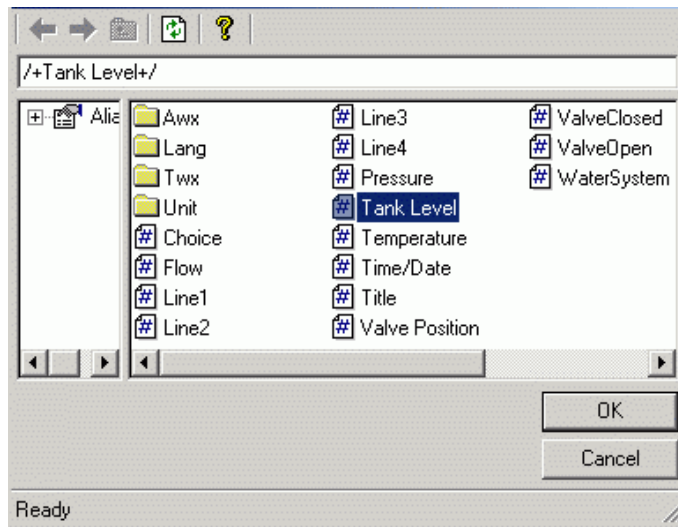


Figure 7.8. Selecting an Alias From the Language Alias Browser

Variables

Variables can be used in expressions. How the variable needs to be referred to depends on the type of variable. A local variable can be used in expressions when the variable is embedded between double tildes.

Local Variables

For local variables within the expression, use the following syntax:

~~local_variable_name~~

Example:

x=~~Setpoint~~

Simulation Variables

For simulation variables within the expression, use the following syntax:

{{simulation_variable_name}}

Example:

x={{gfwsim.random.long}}

Arithmetic

The symbols '+', '-', '*', '/' and '%' use the following format:

expression :: parameter **symbol** parameter

Where

parameter	a local variable, an OPC tag, a constant or another expression
symbol	+ or - or * or / or %

Result:

The expression results in a number of any type (float, long, etc.).

Examples:

Symbol	Description	Example	Result
+	Addition	~~var1~~ + ~~var2~~	8+3=11
-	Subtraction	~~var1~~ - ~~var2~~	8-3=5
*	Multiplication	~~var1~~ * ~~var2~~	8*3=24
/	Division	~~var1~~ / ~~var2~~	8/3=2.66667
%	Calculates the remainder after division	~~var1~~ % ~~var2~~	8%3=2
(and)	Gives precedence to parts of the calculation	~~var1~~ / (~~var2~~ + ~~var3~~)	8/(3+2)=1.6

Relational

The symbols '<','>','<=','>','=','==' and '!=' use the following format:
 expression :: parameter **symbol** parameter

Where

Parameter	A local variable, an OPC tag, a constant or another expression
Symbol	< or > or <= or >= or == or !=

Result:

The expression results in a Boolean value (0 or 1).

Examples:

Symbol	Description	Example	Result
<	Less than	~~var1~~ < ~~var2~~	8<3 = 0
>	Greater than	~~var1~~ > ~~var2~~	8>3 = 1
<=	Less than or equal to	~~var1~~ <= ~~var2~~	8<=3 = 0
>=	Greater than or equal to	~~var1~~ >= ~~var2~~	8>=3 = 1
==	Equal to	~~var1~~ == ~~var2~~	8==3 = 0
!=	Not equal to	~~var1~~ != ~~var2~~	8!=3 = 1

Logical

The symbols '&&' and '||' use the following format:
 expression :: parameter **symbol** parameter

The symbol '!' uses the following format:
 expression :: **symbol** parameter

Where

parameter	A local variable, an OPC tag, a constant or another expression
symbol	&& or or !

Result:

The expression results in a Boolean value (0 or 1).

Truth table:

~~var1~~	0		not 0	
	0	not 0	0	not 0
~~var1~~ && ~~var2~~	0	0	0	1
~~var1~~ ~~var2~~	0	1	1	1
!~~var1~~	1	1	0	0

Examples:

Symbol	Description	Example	Result
&&	And	~~var1~~ && ~~var2~~	8 && 3 = 1
	Or	~~var1~~ ~~var2~~	8 3 = 1
!	Not	!~~var1~~	!8 = 0

Bitwise

The symbols '&', '|', and '^' of the bitwise group use the following format:

expression :: parameter **symbol** parameter

The symbol '~' of the logical group uses the following format:

expression :: **symbol** parameter

The symbols 'shl' and 'shr' of the bitwise group use the following format:

expression :: **symbol** (value, shift by)

Where

parameter	A local variable, an OPC tag, a constant or another expression
symbol	&& or or ^ or shl or shr or ~

Result:

The expression results in a number when the parameters used contain numbers.

Bit table:

	Binary (Decimal)	Binary (Decimal)
~~var1~~	0000.0000.0000.1000 - (8)	0000.0000.0110.0000 - (96)
~~var2~~	0000.0000.0000.1010 - (10)	0000.0000.0000.1000 - (8)
~~var1~~ & ~~var2~~	0000.0000.0000.1000 - (8)	0000.0000.0000.0000 - (0)
~~var1~~ ~~var2~~	0000.0000.0000.1010 - (10)	0000.0000.0110.1000 - (104)
~~var1~~ ^ ~~var2~~	0000.0000.0000.0010 - (2)	0000.0000.0110.1000 - (104)
shl(~~var1~~,3)	0000.0000.0100.0000 - (64)	0000.0011.0000.0000 - (768)
shr(~~var1~~,3)	0000.0000.0000.0001 - (1)	0000.0000.0000.1100 - (12)
~(~~var1~~)	1111.1111.1111.0111 - (-9)	1111.1100.1111.1111 - (-97)
bittest(~~var1~~,3)	0000.0000.0000.0001 - (1)	0000.0000.0000.0000 - (0)

Examples:

Symbol	Description	Example	Result
&	Bit And	~~var1~~ & ~~var2~~	8 & 3 = 0
	Bit Or	~~var1~~ ~~var2~~	8 3 = 11
^	Bit eXclusive Or	~~var1~~ ^ ~~var2~~	8^3=11
shl	Bit shift left	shl(~~var1~~,3)	8<<3=64
shr	Bit shift right	shr(~~var1~~,3)	8>>3=1
~	Not (two's complement)	~(~~var1~~)	!8 = -9
bittest	Bit Test	bittest (5,0)	1

Note: The **bittest** function requires you to specify the position of the bit to be tested. You must indicate that it starts from 0. In other words, a bit position of "0" indicates the "less significant" bit.

Functions

The symbols 'sin', 'asin', 'cos', 'acos', 'tan', 'atan', 'log', 'ln', 'exp', 'sqrt', 'abs', 'ceil', and 'floor' use the following format:

expression :: **symbol** (parameter)

The symbols 'pow', 'min' and 'max' use the following format

expression :: **symbol** (parameter, parameter)

The symbol 'if' uses the following format

expression :: **symbol** (parameter, parameter, parameter)

Where

parameter	A local variable, an OPC tag, a constant or another expression
symbol	Sin, asin, cos, acos, tan, atan, log, ln, exp, sqrt, abs, ceil, floor, min, max, pow, or if

Result:

The expression results in a number.

Examples:

Symbol	Description	Example	Result
sin	Sine of an angle in radians	sin(<code>~~var1~~</code>)	sin(0.785)=0.71
cos	Cosine of an angle in radians	cos(<code>~~var1~~</code>)	cos(0.785)=0.71
tan	Tangent of an angle in radians	tan(<code>~~var1~~</code>)	tan(0.785)=1.0
asin	Arc sine returns an angle in radians	asin(<code>~~var1~~</code>)	asin(0.5)=0.52
acos	Arc cosine returns an angle in radians	acos(<code>~~var1~~</code>)	acos(0.5)=1.05
atan	Arc tangent returns an angle in radians	atan(<code>~~var1~~</code>)	atan(1)=0.785
sqrt	Returns the square root	sqrt(<code>~~var1~~</code>)	sqrt(100)=10
pow	Returns value 1 raised to the power value 2	pow(<code>~~var1~~</code> , <code>~~var2~~</code>)	pow(100,1.5)=1000
log	10 based logarithm	log(<code>~~var1~~</code>)	log(100)=2
ln	e based logarithm	ln(<code>~~var1~~</code>)	ln(7.389)=2
exp	Exponential	exp(<code>~~var1~~</code>)	exp(2)=7.389
abs	absolute value	abs(<code>~~var1~~</code>)	abs(-1)=1
Ceil	integer ceiling	ceil(<code>~~var1~~</code>)	ceil(7.39)=8
Floor	integer floor	floor(<code>~~var1~~</code>)	floor(7.39)=7
Min	lowest value of two	min(<code>~~var1~~</code> , <code>~~var2~~</code>)	min(10,5)=5
Max	highest value of two	max(<code>~~var1~~</code> , <code>~~var2~~</code>)	max(10,5)=10
if	conditional statement	if(<code>~~var1~~ < ~var2~~</code> , <code>~~var1~~</code> , <code>~~var2~~</code>)	if(5<8,5,8)=5
like	Wildcard string compare	Like(string, pattern, casesensitive')	
quality	Quality of tag or expression	See below.	See below.
tostring	Type conversion	See below.	See below.
0x	Hexadecimal constant	x=0x11	17
0t	Octal constant	x=0t11	9
0b	Binary constant	x=0b11	3

Note: For the like operator: "string" equals the string to search in; "pattern" equals the string to search for (can include wildcards); nonzero for case-sensitive search; zero for case-insensitive search. String syntax is "\$string\$".

You can use these special characters in pattern matches in string:

? Any single character.

Zero or more characters.

Any single digit (0-9).

[charlist] Any single character in charlist.

[!charlist] Any single character not in charlist.

Quality

The **quality** option on the **Functions** menu of the **Expression Editor** is used to evaluate the quality of an OPC tag or an expression.

The following general syntax is used for quality expressions:

```
x = quality(expression)
```

Note: The "(expression)" can also be a simple expression composed of a single tag.

The **quality** function returns the OPC quality of the string between parentheses as one of the following results:

192: quality is GOOD

64: quality UNCERTAIN

0: quality BAD

Note: The OPC Foundation establishes the value ranges for quality. There are actually varying degrees of quality:

- GOOD: 192-252
- UNCERTAIN: 64-191
- BAD: 0-63

For more information, refer to the *OPC Data Access Custom Interface Standard* available for download at the OPC Foundation's Web site, <http://www.opcfoundation.org/>

Example Quality Expression

Expression	Result
x=quality({{SMAR.Simulator.1\SimulatePLC.PumpStatus}})	192 (Quality GOOD)

The quality of an expression is determined through the evaluation of each single tag in the expression. Thus, if you have multiple tags in an expression (and each tag has a different quality), the result of the expression (i.e. 192 [GOOD], 64 [BAD], or 0 [UNCERTAIN]) corresponds to the quality of the tag with the lowest quality. If an expression contains a conditional statement (e.g. if, then, or else), then the result of the expression is affected only by the quality of the branch being executed.

Consider the following sample expression:

```
x = if ( quality({{Tag1}}) == 192, {{Tag1}}, {{Tag2}})
```

This expression can be read as follows:

"If the quality of Tag1 is GOOD (i.e. 192), then the expression result (x) is the value of Tag1. In all other cases (i.e. the quality of Tag1 is UNCERTAIN or BAD), the expression result (x) is the value of Tag2."

We can calculate the results for this expression using different qualities for Tag1 and Tag2, as shown in the figure below.

Case	Tag1 quality	Tag2 quality	Result	Result quality
1	GOOD	GOOD	Tag1	192 (GOOD)
2	GOOD	UNCERTAIN	Tag1	192 (GOOD)
3	GOOD	BAD	Tag1	192 (GOOD)
4	UNCERTAIN	GOOD	Tag2	192 (GOOD)
5	UNCERTAIN	UNCERTAIN	Tag2	64 (UNCERTAIN)
6	UNCERTAIN	BAD	Tag2	0 (BAD)
7	BAD	GOOD	Tag2	192 (GOOD)
8	BAD	UNCERTAIN	Tag2	64 (UNCERTAIN)
9	BAD	BAD	Tag2	0 (BAD)

In cases 1-3 above, the quality of Tag1 is GOOD, and therefore the result of the expression is GOOD. Thus, the result of the expression is not affected by the quality of Tag2 (the "else" branch of the expression), which is ignored.

In cases 4-6, the quality of Tag1 is UNCERTAIN, and therefore the result of the expression is the quality of Tag2.

In cases 7-9, the quality of Tag1 is BAD, and therefore the result of the expression is the quality of Tag2.

Note: The "quality()" function returns a value that represents the quality of the expression within the parentheses but is always GOOD_QUALITY. For example, if Tag1 is BAD_QUALITY then the expression "x=quality({{Tag1}})" will return 0 with GOOD_QUALITY.

The result of an expression is the minimum quality of the evaluated tag in the expression and is affected only by the quality of the conditional (if, then, or else) branch that is executed.

Consider the following sample expression:

```
x = if ({{TAG_01}}>0,{{TAG_02}},{{TAG_03}})
```

This expression can be read as follows:

"If the value of TAG_01 is greater than 0, then the expression result (x) is TAG_02. If the value of TAG_01 is less than or equal to 0, then the expression result (x) is TAG_03."

Let's assume that the following values and qualities for these tags:

TAG_01=5 with quality GOOD

TAG_02=6 with quality UNCERTAIN

TAG_03=7 with quality BAD

Because the value of TAG_01 is 5 (greater than 0), the expression result is TAG_02. Thus, the final expression result is 6, and the final expression quality is UNCERTAIN.

Type Conversion

The **tostring** option on the **Functions** menu of the **Expression Editor** takes the value of whatever item is in parentheses and converts it into a string as follows:

The value is +(value)+unit

It can be used to convert from number to string, and it can be very useful for string concatenation.

The proper syntax for the **tostring** option is:

```
x = $"The value is "$ + tostring({{gfwsim.ramp.float}}) + $" unit"$
```

Note: In the expression above, the word "unit" is placeholder text for a user-specified unit of measurement or variable (e.g. Watt, inches, meters, etc.).

Example Expressions Type Conversion

Expression	Result
x = \$"The value is "\$ + tostring({{gfwsim.ramp.float}}) + \$" Watt"\$	"The value is 543.2345152 Watt"

Constants

The **Functions** menu of the **Expression Editor** supports constant values, including hexadecimal, octal, and binary formats.

Example Expressions Using Constants

Expression	Result
x=0x11	17
x=0t11	9
x=0b11	3

The **Expression Editor** conveniently inserts the 0x and 0t and 0b prefixes for you so do not have to recall them.

Interpreting and Translating Constants

The examples below show how values are calculated for each type of constant.

- **Hexadecimal:** $0x20A = 2 * (16^2) + 0 * (16^1) + 10 * (16^0) = 2*256 + 0*16 + 10 * 1 = 512 + 0 + 10 = 522$
- **Octal:** $0t36 = 3 * (7^1) + 6 * (7^0) = 3* 7 + 6* 1 = 21 + 6 = 27$
- **Binary:** $0b110 = 1 * (2^2) + 1 * (2^1) + 0 * (2^0) = 1 * 4 + 1 * 2 + 0 * 1 = 4+2+0 = 6$

Local GraphWorX Variables

To indicate that a data connection is a local variable, use the following syntax:

```
~~local_variable_name~~
```

Selecting **Edit Local Variables** from the **Dynamics** menu opens the **Edit Local Variables** dialog box, shown below. Here you can change the default settings for local variables.

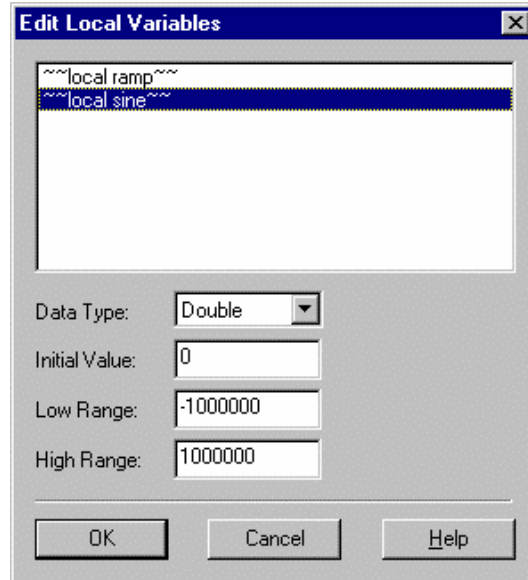


Figure 7.9. Edit Local Variables Dialog Box

Edit Local Variables Parameters

Parameter	Description
Data Type	Specifies the data type of the local variable.
Initial Value	Sets the initial value of the local variable.
High Range and Low Range	Sets a range of values for the local variable.

The scope of local GraphWorX variables is confined to the display in which they are defined. Typically, local variables would be used as values for animation effects (i.e. simulated values that do not need to be defined at the system level).

GraphWorX includes several predefined local simulation variables. You can access a list of these variables by clicking on **Tags Menu** and selecting **Simulation Variables** while in the Property Inspector. This opens the **Simulation Variables dialog**, shown in the figure below, which lists all available simulation variables. These simulation variables are useful for testing display animations when an OPC server is not available.

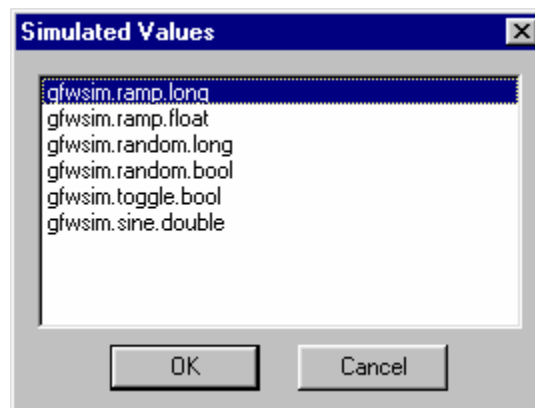


Figure 7.10. GraphWorX Simulated Values

Constant Values

GraphWorX automatically recognizes if an entered data-connection string is a number and interprets that data connection as a constant value. If you want to enter a constant string value (a non-numerical constant), use the following syntax:

```
$"constant_string_value"$
```

This syntax prevents the string from being auto-detected as a tag name.

Aliasing Data Connections

Overview of Aliasing

Aliasing is the ability to define a string (usually a short name) to represent another string (usually part or all of a tag name). You can access the **Edit Aliases** command on the **Dynamics** menu. You can also access it by right-clicking on an object in the display and selecting **Edit Aliases** from the pop-up menu.

When including an alias name in a data connection, the alias name should be surrounded by the special delimiters "<<" and ">>". These delimiters are used to identify the beginning and end of the alias name. For example:

```
[opcserver].<<tank>>.out.
```

In this case "tank" is the alias name.

Second-level aliasing allows an alias name to be entered as the alias definition. This second alias will be resolved prior to the set-aliases command.

Object-Level Aliasing

Objects that utilize data connections can maintain their own alias table. Each dynamic object can have an alias table. This alias table is automatically created when you enter a data source string containing alias names (the table is automatically populated with entries for each alias name used in the data connections of the dynamic object). By default, the alias replacement string is set equal to the alias name. If the alias name is equal to the alias replacement string, the alias will not be resolved. You are only allowed to change the replacement strings in object-level alias tables. The alias names in the table are controlled by the alias names used in the data-source strings.

To edit object-level aliases in GraphWorX, select one or more visible objects and then choose **Edit Aliases** from the **Dynamics** menu. This opens the **Edit Aliases** dialog box. All of the alias tables for all the dynamics attached to all of the selected objects are merged and displayed **Edit Aliases** dialog box. If the selected objects have object names defined, those names will be used to identify which alias names belong to which objects. If no objects are selected when you choose to edit aliases, all aliases in the entire display appear in the **Edit Aliases** dialog box.

Editing Aliases

To open the **Edit Aliases** dialog box, shown below, select **Edit Aliases** from the **Dynamics** menu. This allows you to modify object-level alias tables and to change alias definitions for each alias name entry.

Multiple items in the list control can be selected simultaneously to allow multiple entries to be modified at once. Double-clicking on any alias name automatically selects all duplicate alias names in the list.

To sort the list of aliases by a certain category, simply click the column heading and the list will be arranged according to priorities in that column.

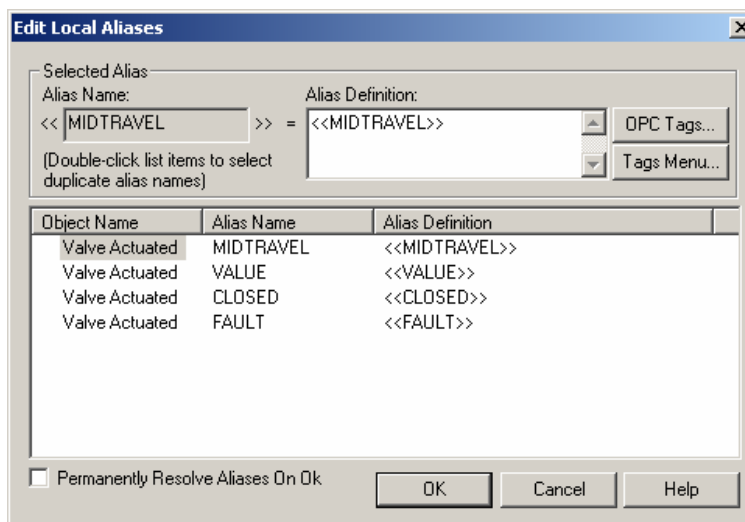


Figure 7.11. Edit Aliases Dialog Box

Edit Aliases Parameters

Parameter	Description
Alias Definition	Changes the definitions for all selected alias names.
OPC Tags button	Launches the OPC Universal Tag Browser .
Tags Menu button	Displays a menu with the following options: Expression Editor , Local Variables , Simulation Variables , and Global Aliases .
Permanently Resolve Aliases on OK check box	Aliases are normally resolved when entering runtime mode (the alias names are restored when you return to configuration mode). Checking this box resolves the aliases in configuration mode (consequently replacing the original aliases permanently).

Note: The alias replacement strings are modifiable during runtime mode via OLE Automation.

Runtime Aliasing

Runtime aliasing encompasses a variety of ways to change a display's aliases during runtime. You can specify a collection of aliases to be set as part of a given command. For example, when using a pick action to load a display, you can specify aliases to be set in the display that is about to be loaded. These aliases will be set before any tags in the new display are requested. The aliases to be set can be specified directly in GraphWorX or can come from an external tab-delimited text file (such files can be exported from Excel, created in Notepad, or created using GraphWorX's Alias File Editor).

Commands with which aliases can be associated include:

- **Pick action-Load display:** Sets the aliases in the newly loaded display.
- **Pick action-Drag/drop load display:** Sets the aliases in the newly drop-loaded display.
- **Pick action-Popup window:** Sets the aliases in the newly popped-up window.
- **Pick action-Embedded window:** Sets the aliases in the newly embedded window.
- **Pick action-Set aliases:** Sets the specified aliases in the currently open display; can apply to the whole display or a specified object or symbol.
- **Pick action-Alias dialog:** Presents the operator with a dialog, and sets the aliases in the current display based on the selections the operator makes in this dialog; can apply to the whole display or a specified object or symbol.
- **Tab Load Display:** Each tab can include a set of aliases.
- **Launching GraphWorX from the command line:** You can specify an alias file as one of the command-line parameters.

To set local aliases, click the **Set Aliases** button on the corresponding pick action tab in the Property Inspector and select **Local Aliases** from the pop-up menu, as shown in the figure below.

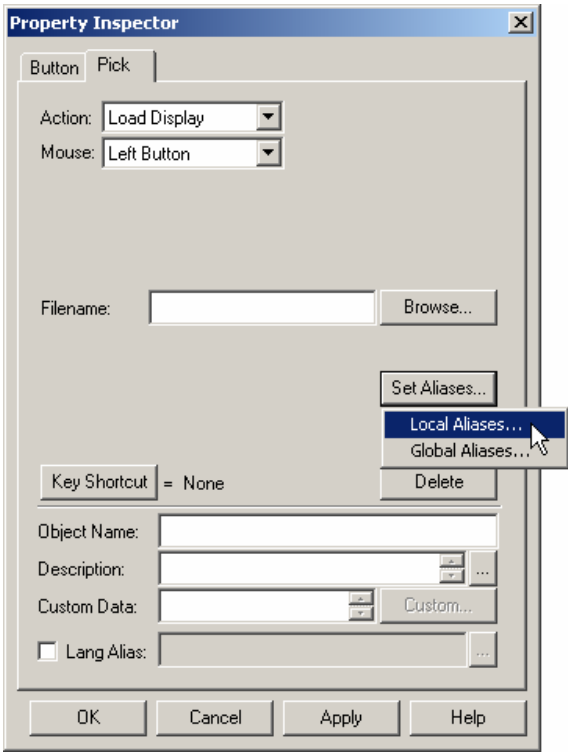


Figure 7.12. Property Inspector: Pick Tab

This opens the **Set Aliases Configuration** dialog box, shown below: This interface is similar to all of the actions, but it may vary slightly depending upon the action being configured.

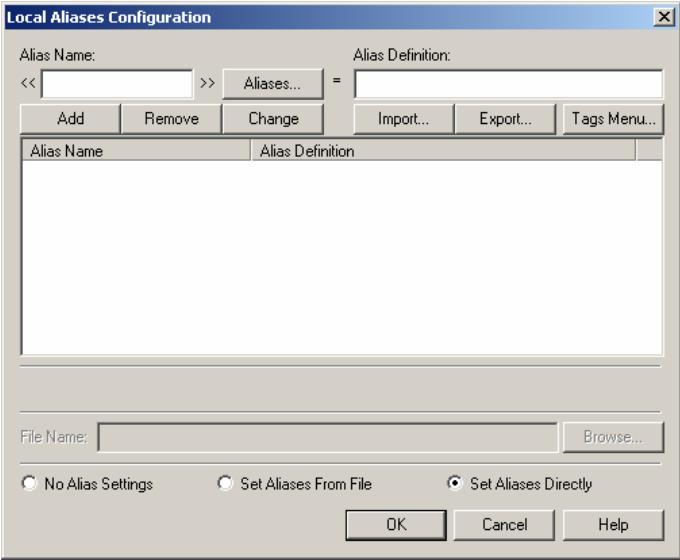


Figure 7.13. Set Aliases Configuration Dialog Box

Set Aliases Parameters

<i>Parameter</i>	<i>Description</i>
Alias Definition	Changes the definitions for all selected alias names.
Tags Menu button	Displays a menu with the following options: OPC Tags , Expression Editor , Local Variables , Simulation Variables , and Global Aliases .
No Alias Settings	If this option is selected, no aliases can be specified.
Set Aliases from File	The aliases can come from an external tab-delimited text file. These files can be exported from Excel, created in Notepad, or created using GraphWorX's Alias File Editor. Click the Browse button to select a file.
Set Alias Directly	The aliases to be set can be specified directly in GraphWorX using the Set Aliases Configuration dialog box.

There are also many OLE Automation methods that correspond to many of the above actions. Refer to the **OLE Automation Reference** section for details.

Note: The alias replacement strings are modifiable during runtime via OLE Automation.

You can also specify global aliases by clicking the **Set Aliases** button on the corresponding pick action tab in the Property Inspector and selecting **Global Aliases** from the pop-up menu.

Select a theme from the **Themes** dialog box, as shown in the figure below. Click **OK**.

Note: To edit a theme's properties, select a theme and click the **Edit** button. This opens the **Theme Editor**, where you can specify theme items and the theme scope. For information about editing global alias themes and specifying the theme scope, please see the **Runtime Advanced Settings** section.

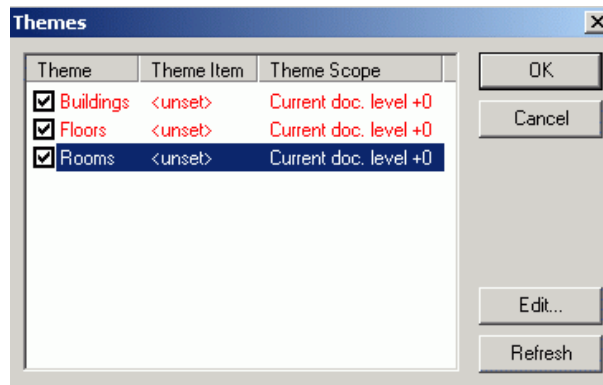


Figure 7.14. Choosing Global Alias Themes

Second-Level Aliasing

Second-level aliasing allows an alias name to be entered as the alias definition. This second alias will be resolved prior to the set-aliases command.

Editing Data Source Connections

Selecting **Edit Connections** from the **Dynamics** menu opens the **Edit Data Source Connection** dialog box, shown in the figure below, which simplifies the connection of GraphWorX symbols to OPC data.

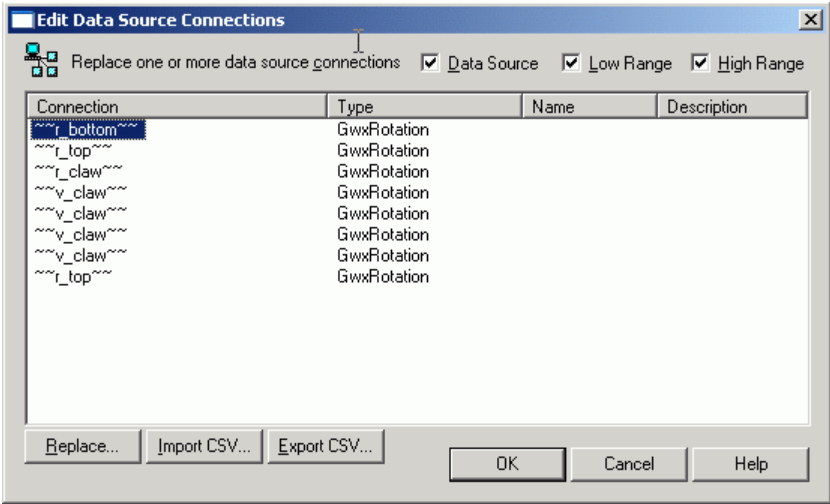


Figure 7.15. Editing the Data Source Connection

Replacing Connections

Clicking the **Replace** button opens the **Replace Connections** dialog box, as shown in the figure below. This allows you to run a search and replace operation for one or more connections on one or more symbols in the current display. The replace capability supports wildcard replacement, similar to Find / Replace feature.

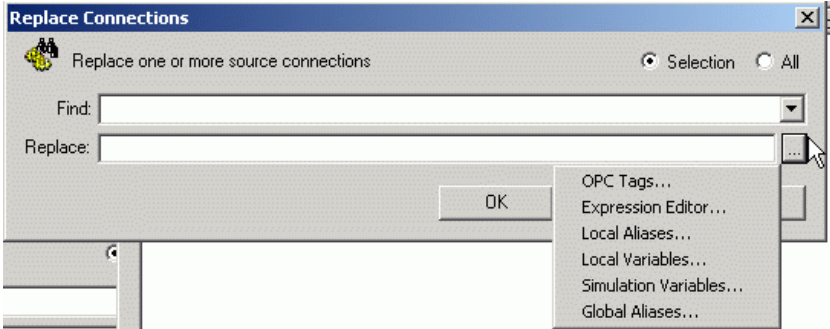


Figure 7.16. Replace Connections Dialog Box

Import and Export

In addition, the tags can be automatically connected to symbols by importing a *.csv file and using it for auto connections. Click the **Import CSV** button to browse for a .csv file.

The existing connections can also be exported to *.csv file for a further use. Click the **Export CSV** button to browse for a .csv file.

Section 8

Dynamic Connections

This chapter explains how to make dynamic connections from display objects to points from various OPC data servers. The objects are animated and controlled by the values of those data points. You can make multiple dynamic connections to a single object. For example you can create an object that changes color and size based on values from two different point connections.

You create dynamic connections with functions contained in the **Dynamics** Menu. Types of dynamic connections include flash, color, process point (PPT), data entry, size, location, analog and digital selector, animator, hide, and rotation.

The Data Source

In every dynamic dialog there is a data source. Refer to the **Data Connections** sections for more information.

Action Dynamics

Dynamic actions connections apply specific actions to objects in a display based on data-point values. You can create these connections through the **Actions** submenu on the **Dynamics** menu, as shown in the figure below, or by clicking the appropriate button on the **Dynamics** toolbar, shown below.

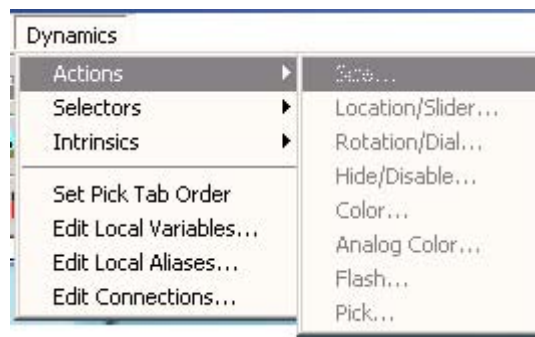


Figure 8.1. Dynamics - Actions Menu



Figure 8.2. Dynamics Toolbar

Size Connection



Using the **Size** button on the **Dynamics** toolbar, you can define an object that changes size based on the real-time value of its connected data point. The object automatically sizes to scale for incoming data points, such that a 100 percent scale equals the full size of the drawn object. You can override the size by defining range override values.

During Runtime Mode, the **Size** connection resizes the object in proportion to the percentage of the high and low ranges of the connected data point.

To establish a size connection:

1. Select an object in the display.
2. Select **Actions > Size** from the **Dynamics** menu, or click the **Size** button on the **Dynamics** toolbar. The **Size** tab of the Property Inspector for the object opens, as shown below.
3. Establish a data connection.
4. Select a size action from the size action parameters listed in the table below.

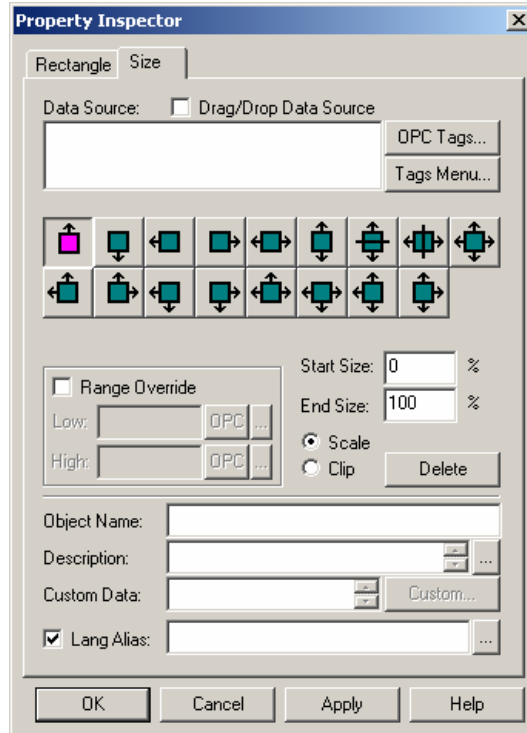


Figure 8.3. Property Inspector Dialog Box: Size Tab

5. Click **OK**. A highlighted box with red handles surrounds the object indicating a dynamic connection, as shown below.

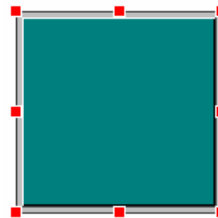


Figure 8.4. Object With Dynamic Connection Established

Size Action Parameters

Parameter	Description
Data Source	Every Dynamic dialog has a data source. Data connections can be typed directly into the Data Source edit control field in the configuration dialogs of the various dynamics. Clicking the OPC Tags button opens the OPC Universal Tag Browser , from which you can select tags in the system. You can also use the options under the Tags Menu to make a data connection. These ways of making data connections for the primary value are also available for making range override data connections.
OPC Tags	Displays the Tag Browser.
Tags Menu	Displays the Tag options: Expression Editor , Aliases , Local Variables , Simulation Variables , and Global Aliases .

Parameter	Description
Size Toolbar	Specifies the direction in which to size the object when the appropriate button is clicked.
Vertical Buttons	Expand the object from the bottom up or top down.
Horizontal Buttons	Expand the object from the left or right.
Bias Bars Buttons	Expand the object from the center above and below the bias line, or to the left and right of the bias line.
Two Directional Button	Expands the object outward from the center.
Three Directional Button	Expands the object in three directions outward from the center.
Range Override	Activates an operating range other than the default range for the data point. Any dynamic based on an analog values, such as process point, or analog selector, will have a range override.
Start Size/End Size	Specifies (as a percentage) the starting and ending size of an object that has changing dimensions. The size does not have to be an integer.
Scale	Scales the object in proportion to the incoming signal.
Clip	Reveals the object.
Delete	Deletes this dynamic. This is only available on actions shown on the Dynamic Actions toolbar.
Object Name	Identifies the object for OLE Automation. A name specified for this object can be referenced to identify this object. For example, this is the object name that can be used in VBA.
Description	Describes the object and usually coincides with what is displayed under ToolTips. Note: You can also select either Global Aliases or Language Aliases from the alias browsers by clicking the ... button to the right of the Description field.
Custom Data	Customized configured data. For example, this could be a custom configured data string. This will be enabled through the custom configuration utility tied to the Custom button. Refer to the Custom Configuration section for details.
Language Alias	Displays a language-aliased string (if applicable). Checking the Lang Alias check box and clicking the ... button next to the Lang Alias box opens the Language Alias Browser , where you can select from all language aliases in the database.

Location/Slider



The **Location/Slider** dynamic on the **Dynamics** toolbar enables you to change the location of an object based on the value of a data point. During runtime mode, the object moves within a specified distance (horizontally, vertically, diagonally, or along a specified path) based on the value of the connected data point.

The **Slider** function emulates a mechanical slider control. During runtime mode, you can move the slider knob to change the value of the connected data point.

To make a location connection:

1. Select an object in the display.
2. Select **Actions > Location/Slider** from the **Dynamics** menu, or click the **Location/Slider** button on the **Dynamics** toolbar. The **Location** tab of the Property Inspector for the object opens, as shown below.

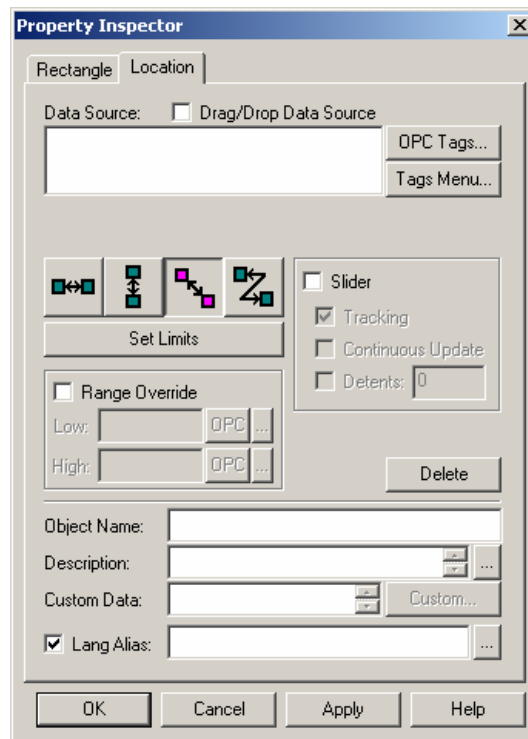


Figure 8.5. Property Inspector Dialog Box: Location Tab

3. Specify the data source tag and select the location/slider action parameters that are described in the table below.
4. Click **OK**. Red handles appear around the object, indicating that a dynamic connection has been made.

Location Action Parameters

Parameter	Description
Data Source	Every Dynamic dialog has a data source. Data connections can be typed directly into the Data Source edit control field in the configuration dialogs of the various dynamics. Clicking the OPC Tags button opens the OPC Universal Tag Browser , from which you can select tags in the system. You can also use the options under the Tags Menu to make a data connection. These ways of making data connections for the primary value are also available for making range override data connections.
OPC Tags	Displays the Tag Browser.
Tags Menu	Displays the Tag options: Expression Editor, Aliases, Local Variables, Simulation Variables, and Global Aliases.
Side to Side Button	Object moves from side to side.
Up and Down Button	Object moves up and down.
Diagonal Button	Object moves diagonally.
Free-Form Path Button	Object moves along a user-specified free-form path.
Range Override	Activates an operating range other than the default range for the data point.
Set limits	Defines an area within which the object will move. A gray frame around the object indicates that the object in runtime will move within that area.
Slider	The slider allows you to hold the object with the mouse and move the object.
Tracking	This is part of the slider action. If this option is checked, the object will move based on the connected data value. If it is not checked, the object will remain stationary, unless you move it manually.
Delete	Deletes this dynamic.

Parameter	Description
Detents	Also a part of the slider action. The range of the variable, tag, or expression that the slider is controlling will be divided into <i>N</i> steps including the high and low range, where <i>N</i> is the value entered in the Detents field.
Object Name	Identifies the object for OLE Automation. A name specified to this object can be referenced to identify this object. For example, this is the object name that can be used in VBA.
Description	Describes the object and usually coincides with what is displayed under ToolTips. Note: You can also select either Global Aliases or Language Aliases from the alias browsers by clicking the ... button to the right of the Description field.
Custom Data	Customized configured data. For example, this could be a custom configured data string. This will be enabled through the custom configuration utility tied to the Custom button. Refer to the Custom Configuration section for details.
Language Alias	Displays a language-aliased string (if applicable). Checking the Lang Alias check box and clicking the ... button next to the Lang Alias box opens the Language Alias Browser , where you can select from all language aliases in the database.

Rotation/Dial



The **Rotation/Dial** dynamic on the **Dynamics** toolbar enables you to rotate an object based on the connected data point value. You define the rotation pivot coordinates. This function is useful when you want to graphically display information on dials or meters in your display.

To define rotation for an object:

1. Select the object.
2. Select **Actions > Rotation/Dial** from the **Dynamics** menu, or click the **Rotation/Dial** button on the **Dynamics** toolbar.
3. The **Rotation** tab of the Property Inspector for the object opens, as shown below.
4. Select the desired parameters described in the table below, and then click **OK**.

Note: It is now possible to freely rotate any image in GraphWorX both in configuration mode using the “free rotate” tool and in runtime mode using the rotation dynamic.

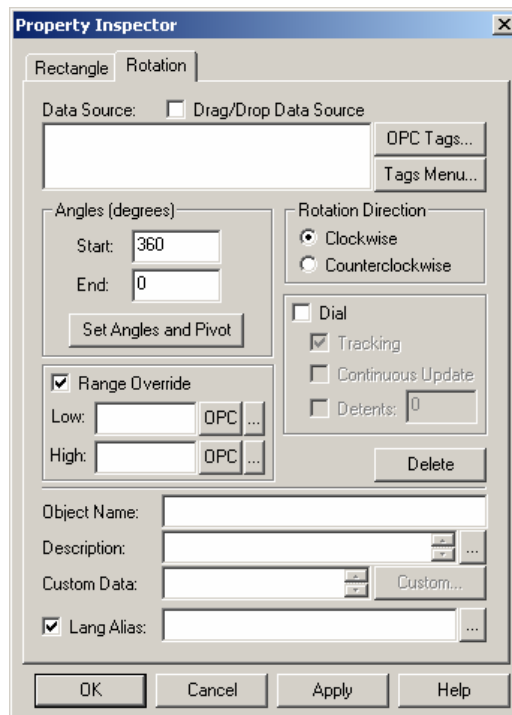


Figure 8.6. Property Inspector Dialog Box: Rotation Tab

Rotation Parameters

Parameter	Description
Data Source	Every Dynamic dialog has a data source. Data connections can be typed directly into the Data Source edit control field in the configuration dialogs of the various dynamics. Clicking the OPC Tags button opens the OPC Universal Tag Browser , from which you can select tags in the system. You can also use the options under the Tags Menu to make a data connection. These ways of making data connections for the primary value are also available for making range override data connections.
OPC Tags	Displays the Tag Browser.
Tags Menu	Displays the Tag options: Expression Editor , Aliases , Local Variables , Simulation Variables , and Global Aliases .
Start	Sets the angle at which the rotation starts.
End	Sets the angle at which the rotation ends.
Set Angles and Pivot	Sets the angle of rotation and defines the coordinates of the pivot point of the object from which it rotates.
Rotation Direction	Specifies a rotation direction (clockwise or counterclockwise).
Dial	Similar to the Slider function described for the Location action. However, instead of allowing you to manually move the object up and down, Dial allows you to manually rotate the object.
Tracking	This is part of the dial action. If this option is checked, the object moves based on the connected data value. If it is not checked, the object remains stationary, unless you move it manually.
Detents	Also a part of the dial action. The range of the variable, tag, or expression that the slider is controlling is divided into N steps including the high and low range, where N is the value entered in the Detents field.
Range Override	Activates an operating range other than the default range for the data point.
Delete	Deletes this dynamic.
Object Name	Identifies the object for OLE Automation. A name specified to this object can be referenced to identify this object. For example, this is the object name that can be used in VBA.
Description	Describes the object and usually coincides with what is displayed under ToolTips. Note: You can also select either Global Aliases or Language Aliases from the alias browsers by clicking the ... button to the right of the Description field.
Custom Data	Customized configured data. For example, this could be a custom configured data string. This will be enabled through the custom configuration utility tied to the Custom button. Refer to the Custom Configuration section for details.
Language Alias	Displays a language-aliased string (if applicable). Checking the Lang Alias check box and clicking the ... button next to the Lang Alias box opens the Language Alias Browser , where you can select from all language aliases in the database.

Hide/Disable

The **Hide/Disable** dynamic on the **Dynamics** toolbar causes objects to become hidden when the value of the digital data point to which it is connected goes either true (logical 1) or false (logical 0) as you configure it.

To make a hide connection:

1. Select an object in the display.
2. Select **Actions > Hide/Disable** from the **Dynamics** menu, or click the **Hide/Disable** button on the **Dynamics** toolbar.
3. The **Hide** tab of the Property Inspector for the object opens, as shown below.
4. Establish a digital data-point connection.
5. Select to hide or disable the object.
6. Select to hide the object when the data-point value is true or when the data-point value is false. Then click **OK**.

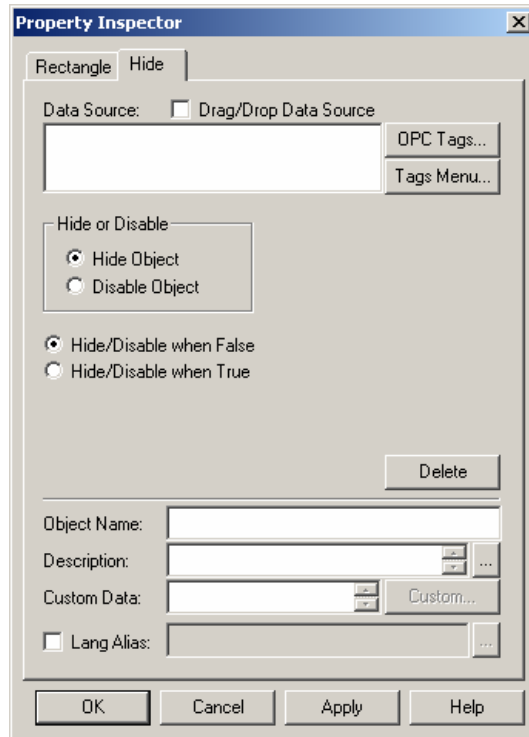


Figure 8.7. Property Inspector Dialog Box: Hide Tab

Hide Parameters

Parameter	Description
Data Source	Every Dynamic dialog has a data source. Data connections can be typed directly into the Data Source edit control field in the configuration dialogs of the various dynamics. Clicking the OPC Tags button opens the OPC Universal Tag Browser , from which you can select tags in the system. You can also use the options under the Tags Menu to make a data connection. These ways of making data connections for the primary value are also available for making range override data connections.
OPC Tags	Displays the Tag Browser.
Tags Menu	Displays the Tag options: Expression Editor , Aliases , Local Variables , Simulation Variables , and Global Aliases .
Hide Object	Hides the selected object in the display.
Disable Object	Disables the selected object in the display.
Hide/Disable When False	Causes the objects to become hidden when the value of the digital data point to which it is connected goes to false (logical 0).
Hide/Disable When True	Causes the objects to become disabled when the value of the digital data point to which it is connected goes to true (logical 1).
Delete	Deletes this dynamic.
Object Name	Identifies the object for OLE Automation. A name specified to this object can be referenced to identify this object. For example, this is the object name that can be used in VBA.
Description	Describes the object and usually coincides with what is displayed under ToolTips. Note: You can also select either Global Aliases or Language Aliases from the alias browsers by clicking the ... button to the right of the Description field.
Custom Data	Customized configured data. For example, this could be a custom configured data string. This will be enabled through the custom configuration utility tied to the Custom button. Refer to the Custom Configuration section for details.
Language Alias	Displays a language-aliased string (if applicable). Checking the Lang Alias check box and clicking the ... button next to the Lang Alias box opens the Language Alias Browser , where you can select from all language aliases in the database.

Digital Color



The **Digital Color** dynamic on the **Dynamics** toolbar allows the color of the object to change based on a digital event. The color connections are prioritized in the order in which they are created. Therefore, if two data points are true at the same time, the higher-priority color connection takes precedence. If none of the connected data points is true, the object is displayed in its original color.

The **Color** connection is used to indicate states such as alarm conditions and temperature changes, or to notify operators of the flow in a pipe.

To create a digital color connection:

1. Select an object in the display.
2. Select **Actions > Color** from the **Dynamics** menu, or click the **Color** button on the **Dynamics** toolbar. The **Color** tab of the Property Inspector for the object opens, as shown below.
3. Select the desired parameters, and then click **OK**.

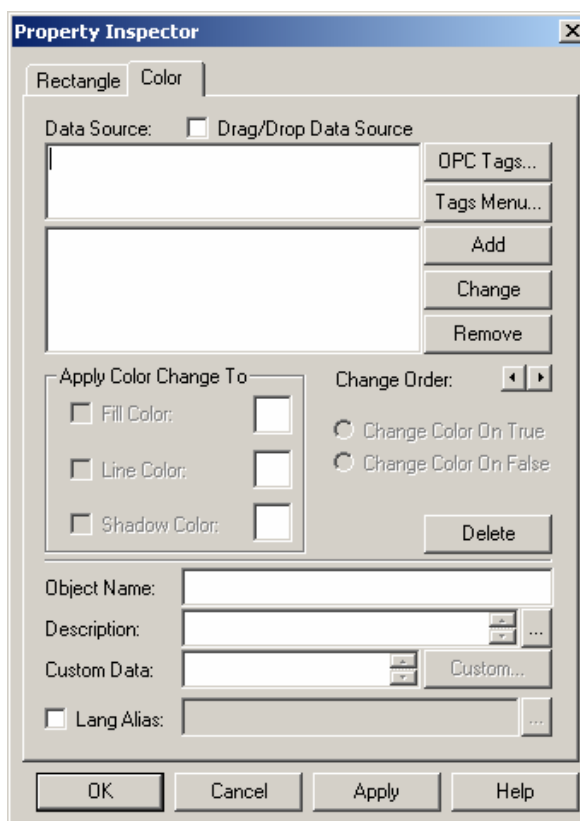


Figure 8.8. Property Inspector Dialog Box: Color Tab

Color Connection Parameters

Parameter	Description
Data Source	Every Dynamic dialog has a data source. Data connections can be typed directly into the Data Source edit control field in the configuration dialogs of the various dynamics. Clicking the OPC Tags button opens the OPC Universal Tag Browser , from which you can select tags in the system. You can also use the options under the Tags Menu to make a data connection. These ways of making data connections for the primary value are also available for making range override data connections.
OPC Tags	Displays the Tag Browser.
Tags Menu	Displays the Tag options: Expression Editor , Aliases , Local Variables , Simulation Variables , and Global Aliases .

Parameter	Description
Add Button	Adds the string currently in the data source edit control to the list of data sources. Click the arrows next to Change Order to sort the data sources.
Change Button	Changes existing data connections. The currently selected item in the data source list box is changed to the string that is currently in the data source edit control.
Remove Button	Removes existing data connections from the data source list box.
Apply Color Change To	These items vary depending on which source string is selected.
Fill Color	If checked, changes the fill color of the object. Clicking on the box to the right of the Fill Color check box opens the color palette.
Line Color	Changes the color of the object's border. Clicking on the box to the right of the Line Color check box opens the color palette.
Shadow Color	Changes the color of the object's shadow. Clicking on the box to the right of the Shadow Color check box opens the color palette.
Change Color on True/False	Changes the color of the object when the connected data point goes true (logical 1) or false (logical 0).
Delete	Deletes this dynamic.
Object Name	Identifies the object for OLE Automation. A name specified to this object can be referenced to identify this object. For example, this is the object name that can be used in VBA.
Description	Describes the object and usually coincides with what is displayed under ToolTips. Note: You can also select either Global Aliases or Language Aliases from the alias browsers by clicking the ... button to the right of the Description field.
Custom Data	Customized configured data. For example, this could be a custom configured data string. This will be enabled through the custom configuration utility tied to the Custom button. Refer to the Custom Configuration section for details.
Language Alias	Displays a language-aliased string (if applicable). Checking the Lang Alias check box and clicking the ... button next to the Lang Alias box opens the Language Alias Browser , where you can select from all language aliases in the database.

Analog Color



The **Analog Color** dynamic on the **Dynamics** toolbar changes the start color and end color based on an analog signal.

Note: You can attach multiple analog color dynamics to shift between more than two colors.

To create an analog color connection:

1. Select an object in the display.
2. Select **Actions > Analog Color** from the **Dynamics** menu. The **Analog Color** tab of the Property Inspector for the object opens, as shown below:
3. Select the desired parameters described in the table below, and then click **OK**.

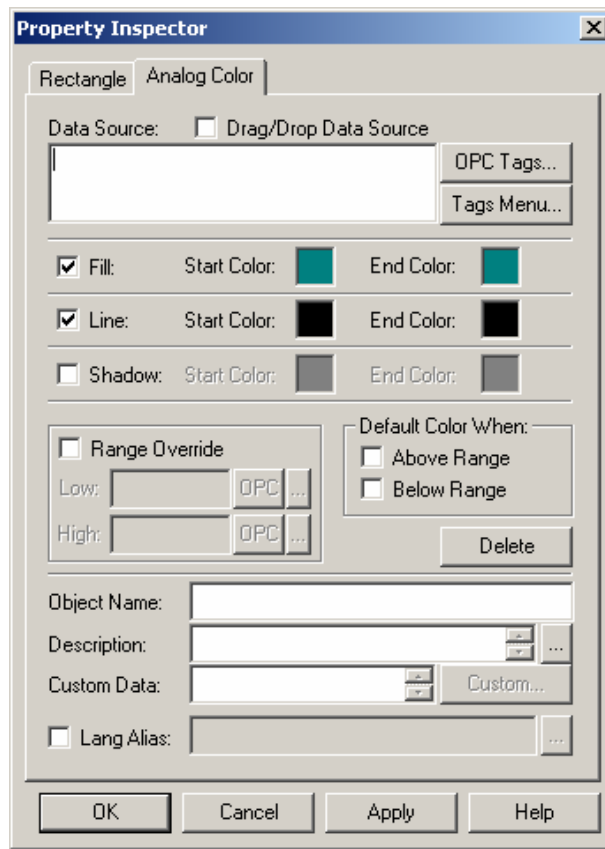


Figure 8.9. Property Inspector Dialog Box: Analog Color Tab

Analog Color Parameters

Parameters	Description
Data Source	Every Dynamic dialog has a data source. Data connections can be typed directly into the Data Source edit control field in the configuration dialogs of the various dynamics. Clicking the OPC Tags button opens the OPC Universal Tag Browser , from which you can select tags in the system. You can also use the options under the Tags Menu to make a data connection. These ways of making data connections for the primary value are also available for making range override data connections.
OPC Tags	Displays the Tag Browser.
Tags Menu	Displays the Tag options: Expression Editor , Aliases , Local Variables , Simulation Variables , and Global Aliases .
Fill Color - Start/End	Changes the starting and ending fill colors of the object. Clicking on the Start Color and End Color boxes opens the Color Palette.
Start Color	Changes the starting fill color of the object. This is the color that will appear when the analog signal is at its lowest value.
End Color	Changes the ending fill color of the object. This is the color that will appear when the analog signal is at its highest value. When the analog signal is between its minimum and maximum values, the color will be a mix of the start and end colors.
Line Color - Start/End	Changes the starting and ending line colors of the object.
Shadow Color - Start/End	Changes the starting and ending shadow colors of the object.
Range Override: Low/High	Activates an operating range other than the default with specific options of choosing low range and/or high range.
Default Color when Above Range/Below Range	Sets the default color when above or below the specified range. The default color is the color of the object to which this dynamic is attached.
Delete	Deletes this dynamic.

Parameters	Description
Object Name	Identifies the object for OLE Automation. A name specified to this object can be referenced to identify this object. For example, this is the object name that can be used in VBA.
Description	Describes the object and usually coincides with what is displayed under ToolTips. Note: You can also select either Global Aliases or Language Aliases from the alias browsers by clicking the ... button to the right of the Description field.
Custom Data	Customized configured data. For example, this could be a custom configured data string. This will be enabled through the custom configuration utility tied to the Custom button. Refer to the Custom Configuration section for details.
Language Alias	Displays a language-aliased string (if applicable). Checking the Lang Alias check box and clicking the ... button next to the Lang Alias box opens the Language Alias Browser , where you can select from all language aliases in the database.

Flash



The **Flash** dynamic on the **Dynamics** toolbar causes an object with a digital connection to blink on and off or to flash between two colors during runtime mode when the connected digital value goes true (logical 1) or false (logical 0).

The default operation is for the object to flash between configured colors. The Property Inspector shows the default as on/off. When the data-point value is false (logical 0), the object displays in the normally on/normally off setting. When the value is true (logical 1), the object blinks between the colors or between visible and hidden. The rate at which the object blinks is determined by the **Flash Rate** value.

To make a flashing connection:

1. Select an object in the display.
2. Select **Actions > Flash** from the **Dynamics** menu, or click the **Flash** button on the **Dynamics** toolbar.
3. The **Flash** tab of the Property Inspector for the object opens, as shown below.
4. Establish a data connection and select the desired parameters described in the table below.
5. Click **OK** when you are finished.

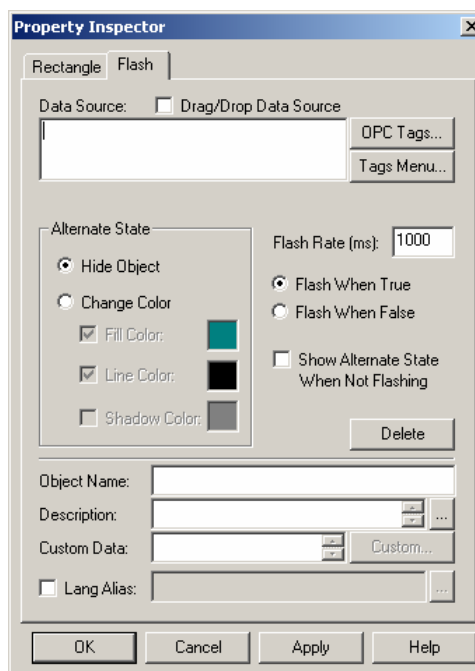


Figure 8.10. Property Inspector Dialog Box: Flash Tab

Flashing Parameters

Parameter	Description
Data Source	Every Dynamic dialog has a data source. Data connections can be typed directly into the Data Source edit control field in the configuration dialogs of the various dynamics. Clicking the OPC Tags button opens the OPC Universal Tag Browser , from which you can select tags in the system. You can also use the options under the Tags Menu to make a data connection. These ways of making data connections for the primary value are also available for making range override data connections.
OPC Tags	Displays the Tag Browser.
Tags Menu	Displays the Tag options: Expression Editor , Aliases , Local Variables , Simulation Variables , and Global Aliases .
Flash Rate	The frequency of the flash in milliseconds. The minimum flash value is 1000 ms.
Flash When True	Flashes when the data-point value is true (logical 1). The object blinks between the colors or between visible and hidden.
Flash When False	Flashes when the data-point value is false (logical 0).
Hide Object	Selects the alternate state as a hidden object.
Change color	Selects the alternate state as a color change.
Fill Color	Selects the color fill for the object. Clicking on the box at right opens the Color Palette.
Line Color	Selects the line color for the object. Clicking on the box at right opens the Color Palette.
Shadow Color	Selects the shadow color for the object. Clicking on the box at right opens the Color Palette.
Show Alternate State When Not Flashing	When not flashing, the object is shown in the specified alternate color, or the object is hidden if that is the chosen alternate state.
Delete	Deletes this dynamic.
Object Name	Identifies the object for OLE Automation. A name specified to this object can be referenced to identify this object. For example, this is the object name that can be used in VBA.
Description	Describes the object and usually coincides with what is displayed under ToolTips. Note: You can also select either Global Aliases or Language Aliases from the alias browsers by clicking the ... button to the right of the Description field.
Custom Data	Customized configured data. For example, this could be a custom configured data string. This will be enabled through the custom configuration utility tied to the Custom button. Refer to the Custom Configuration section for details.
Language Alias	Displays a language-aliased string (if applicable). Checking the Lang Alias check box and clicking the ... button next to the Lang Alias box opens the Language Alias Browser , where you can select from all language aliases in the database.

Pick Actions

The **Pick Action** function allows you to configure an object such that it performs an action when the object is clicked during runtime mode. Such actions include loading displays, popping up windows, launching applications, and downloading values.

To make a pick action connection:

1. Select an object in the display.
2. Select **Actions > Pick** from the **Dynamics** menu, or click the **Pick** button on the **Dynamics** toolbar.
3. The **Pick** tab of the Property Inspector for the object opens, as shown below.

Note: the appearance of the **Pick** tab varies depending upon which action is selected.

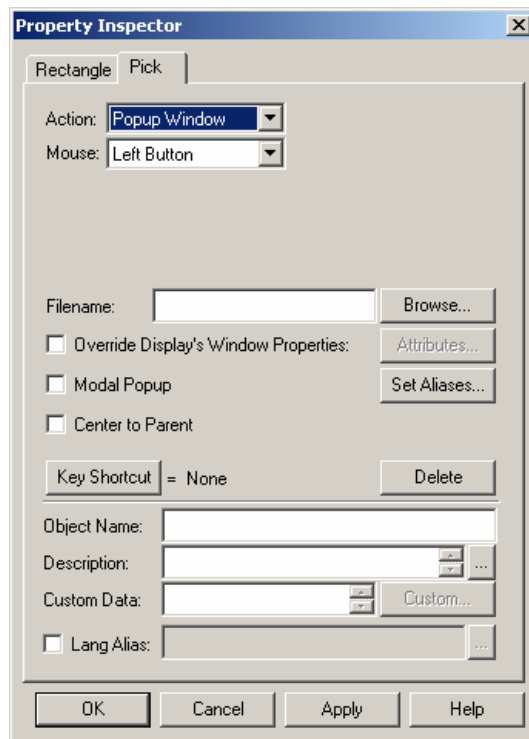


Figure 8.11. Property Inspector Dialog Box: Pick Action Tab

4. Select the desired parameters from the combo box in the **Action** field.

Clicking on the **Key Shortcut** button opens the **Define Key Shortcut** dialog box, shown below. Enter a key and/or choose a modifier, and then click **OK**. When the display is changed to runtime mode, the operator can perform the pick action by pressing the designated shortcut key.

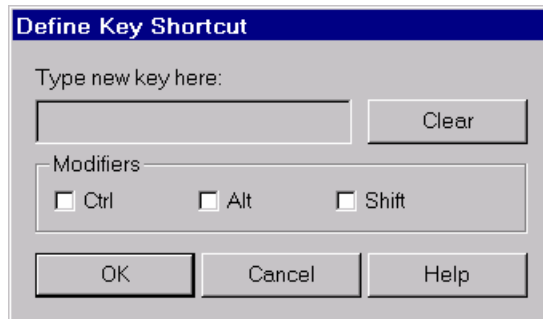


Figure 8.12. Define Key Shortcut Dialog Box

Parameters for Pick Action

Pick actions are listed in the **Action** field in the **Pick** tab of the Property Inspector. Although the parameters for each pick action vary, many of the various pick actions available for pick dynamics have common properties, which are described in the table below.

Pick Action Parameters

Parameter	Description
Mouse	Lets you choose which mouse button will activate the pick during runtime mode (can be left, middle, or right mouse button).
Delete	Deletes this dynamic.
Pick	There are two different "picks": Object pick and three styles of buttons: Normal, Check Box, and Radio.
Check Button	This is a check box. Most of the parameters are the same as the Normal button.

Parameter	Description
Radio Button	When this pick is selected, any other picks that are below the same group are automatically deselected (i.e. they are mutually exclusive).
Initial state	Available only for check button and radio button types. Lets you choose an initial state, which can be up or down.
Group	Available only for the radio button type, this specifies the group name to which the radio button belongs.
Execution Trigger	Selects one of three options for executing the action; On Down, While Down, or On Up.
On Down	Executes the action when the mouse button is clicked down.
On Up	Executes the action when the mouse button is released.
While Down	Executes the action while the mouse button is held down. Lets you specify an interval in milliseconds.
Key Shortcut	Defines a shortcut key for the pick action.
Object Name	Identifies the object for OLE Automation. A name specified to this object can be referenced to identify this object. For example, this is the object name that can be used in VBA.
Description	Describes the object and usually coincides with what is displayed under ToolTips. Note: You can also select either Global Aliases or Language Aliases from the alias browsers by clicking the ... button to the right of the Description field.
Custom Data	Customized configured data. For example, this could be a custom configured data string. This will be enabled through the custom configuration utility tied to the Custom button. Refer to the Custom Configuration section for details.
Language Alias	Displays a language-aliased string (if applicable). Checking the Lang Alias check box and clicking the ... button next to the Lang Alias box opens the Language Alias Browser , where you can select from all language aliases in the database.

Action - Load Display

The **Load Display** pick action in the **Pick** tab of the Property Inspector loads the display in runtime mode. You can also configure aliases by clicking the **Set Aliases** button.

Load Display Parameters

Parameter	Description
File Name	Click the Browse button to pick a file name (the display file to be loaded during runtime).

Action - Drag/Drop Load

The **Drag/Drop Load** pick action in the **Pick** tab of the Property Inspector allows you to drag-and-drop a load display to another GraphWorX window (not on itself or in a parent window). This provides an easy mechanism for choosing into which window the display gets loaded.

Drag/Drop Load Parameters

Parameter	Description
File Name	Click the Browse button to pick a file name (the display file to be loaded during runtime).

Action - Display Back and Display Forward

GraphWorX maintains a history of the last 50 displays. You can navigate through the display history using the pick actions **Display Back** and **Display Forward** in the **Pick** tab of the Property Inspector. These commands are also available as pick actions and as OLE Automation methods. The Display History feature is similar to MS Internet Explorer's back/forward feature.

Note: The display file history (forward/back commands) remembers the initial alias settings specified when a display is opened and will reset those aliases when you go back to that display in the file history. Refer to the **Runtime Aliasing** section for more information.

Action - Pop-Up Window

The **PopUp Window** pick action in the **Pick** tab of the Property Inspector opens a popup GraphWorX window in runtime mode.

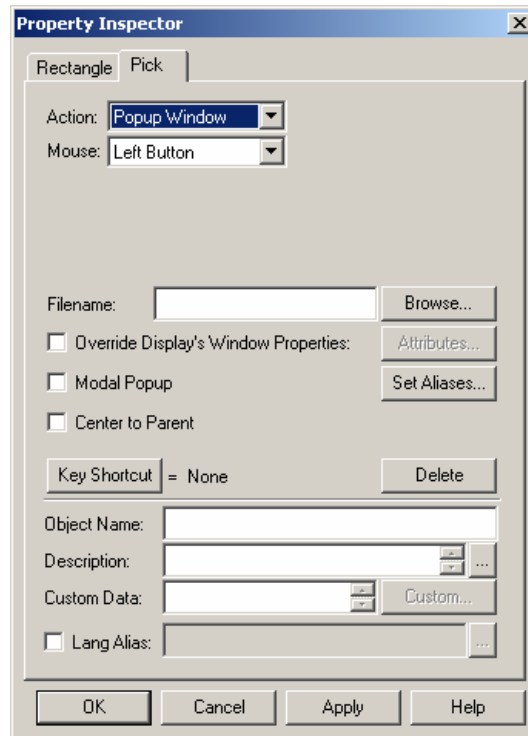


Figure 8.13. Property Inspector Dialog Box: Popup Window Action

Popup Window Parameters

Parameter	Description
File Name	Click the Browse button to pick a file name (the display file to be loaded into the popup window during runtime).
Override Display's Window Properties	Allows you to override the window properties of the display to be loaded.
Modal Popup	Opens the popup window as a modal window (i.e. disables the parent window).
Center to Parent	Automatically centers the popup window to the parent window when opened.
Attributes	Checking the Override Display's Window Properties check box enables the Attributes button. Clicking on this button opens the Popup Window Properties dialog box, which allows you to specify the window's dimensions and other attributes.

Action - Embedded Window

Embedded Window pick action in the **Pick** tab of the Property Inspector is similar to the regular **Popup Window** pick action except that an embedded window is a "child" window of the display that launched the embedded window. The embedded window can move, scroll, and scale with the parent display. The parameter **Modal Popup** of the **Popup Window** action is not supported by the **Embedded Window** action.

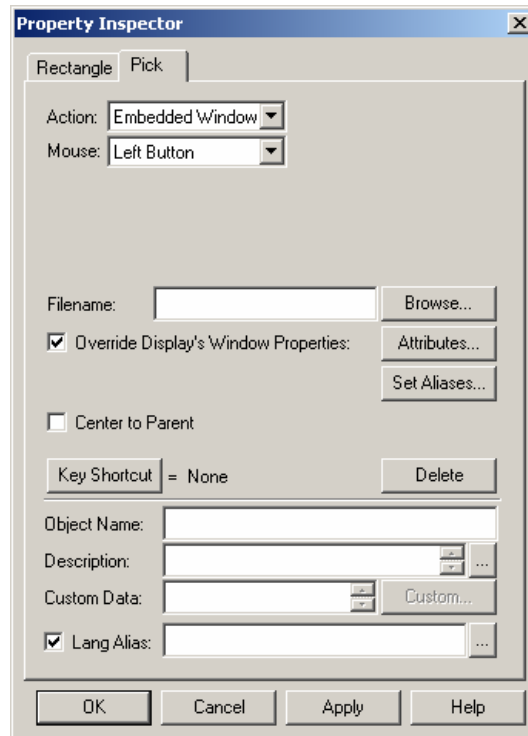


Figure 8.14. Property Inspector Dialog Box: Embedded Window Action

Embedded Window Parameters

Parameter	Description
File Name	Click the Browse button to pick a file name (the display file to be loaded into the popup window during runtime).
Override Display's Window Properties	Overrides the window properties of the display to be loaded.
Center to Parent	Automatically centers the popup window to the parent window when opened.
Attributes	Checking the Override Display's Window Properties check box enables the Attributes button. Clicking on this button opens the Popup Window Properties dialog box, which allows you to specify the window's dimensions and other attributes.

Action - Launch Application

The **Launch Application** pick action in the **Pick** tab of the Property Inspector launches the specified application when GraphWorX is in runtime mode. It provides the **Delete**, **Key Shortcut**, and **Mouse** parameters that have been discussed previously.

In addition to launching executables, you can launch various documents, including Microsoft Word documents, Microsoft Excel spreadsheets, HTML files, and text files. The registered application is automatically started when the file is launched.

Launch Application Parameters

Parameter	Description
File Name	Click the Browse button to pick an executable or document name (the file to be loaded during runtime).

Action - Launch Web Page

The **Launch Web Page** pick action in the **Pick** tab of the Property Inspector, shown in the figure below, launches a specified URL or Web site when GraphWorX is in runtime mode. Simply enter the URL address in the **Web Page** field, as shown in the figure below. When the action is executed in runtime mode, the specified Web page opens in the default Web browser.

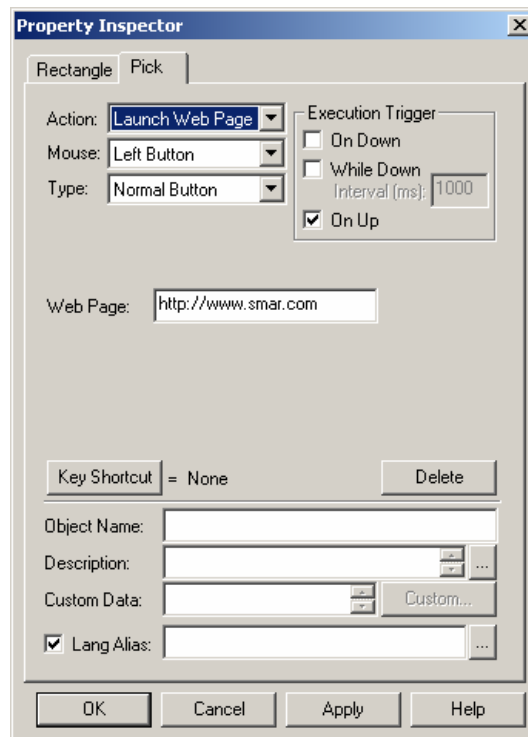


Figure 8.15. Property Inspector Dialog Box: Launch Web Page Action

If you set this pick action for a push button, the button is automatically named "Go to" followed by the specified URL address, as shown in the figure below.



Figure 8.16. Push Button With Launch Web Page Pick Action

Action - Close Window

The **Close Window** pick action in the **Pick** tab of the Property Inspector closes the window to which this pick belongs. It provides the **Delete**, **Key Shortcut**, and **Mouse** parameters that have been discussed previously.

Action - Toggle Value

The **Toggle Value** pick action in the **Pick** tab of the Property Inspector is used to toggle between two specified values during runtime mode.

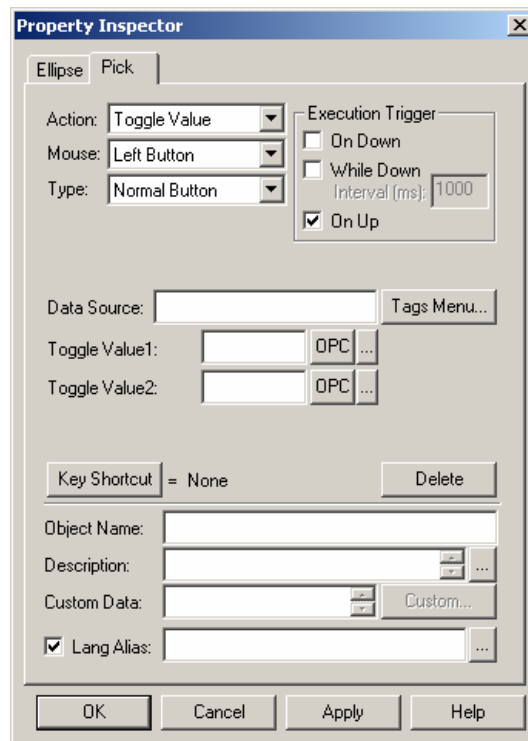


Figure 8.17. Property Inspector Dialog Box: Toggle Value Action

Toggle Value Parameters

Parameter	Description
Data Source	The data source is the tag to which the downloaded values are to be written. Click the Tags Menu button to quickly insert the desired OPC tag, alias, local variable, simulation variable, or global alias.
Toggle Value 1	Specifies the first toggle value when the button is clicked. The downloaded value can be a constant value, the value of a tag, the calculated value of an expression, an alias, or a global alias.
Toggle Value 2	Specifies the second toggle value when the button is clicked. The downloaded value can be a constant value, the value of a tag, the calculated value of an expression, an alias, or a global alias.

Action - Download Value

The **Download Value** pick action in the **Pick** tab of the Property Inspector is used to download values when you click on the pick object.

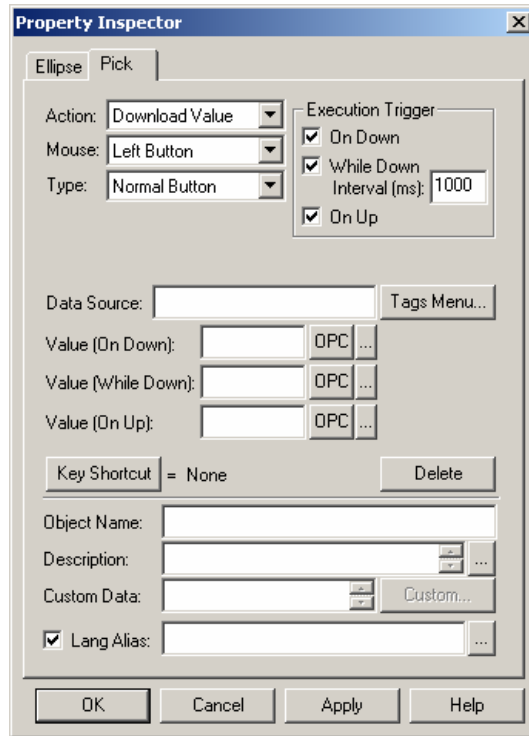


Figure 8.18. Property Inspector Dialog Box: Download Value Action

Download Value Parameters

Parameter	Description
Data Source	The data source is the tag to which the downloaded values are to be written. Click the Tags Menu button to quickly insert the desired OPC tag, alias, local variable, simulation variable, or global alias.
Values (On Down, While Down, On Up)	Specifies the values to be downloaded. The downloaded value can be a constant value, the value of a tag, the calculated value of an expression, an alias, or a global alias. Note: If no value is specified for the On-Up event, the pick action will display a number pad or a keypad dialog in runtime, where you can enter the value. This feature must be manually enabled in the Runtime Advanced tab of the Application Preferences dialog box by checking the Ask for Download Value check box.

Action - Set Aliases

The **Set Aliases** pick action in the **Pick** tab of the Property Inspector sets aliases for data connections in runtime mode. Click the **Set Aliases** button on the property inspector to open the **Set Aliases Configuration** dialog box. Refer to the section **Aliasing Data Connections** for details.

Action - Aliases Dialog

This **Alias Dialog** pick action in the **Pick** tab of the Property Inspector displays the Aliases dialog for the display objects or for the whole display in runtime mode. Click the **Set Aliases** button on the Property Inspector to open the **Set Aliases Configuration** dialog box. Refer to the section **Aliasing Data Connections** for details.

Action - Run Script

The **Run Script** pick action in the **Pick** tab of the Property Inspector runs three different types of scripts: VBA, VBScript and JScript.

1. Select the type from the **Script Type** drop-down list, as shown in the figure below.

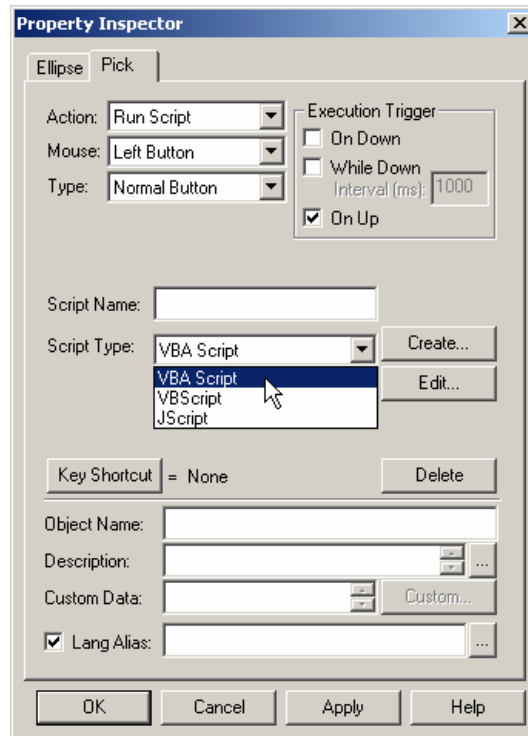


Figure 8.19. Choosing a Script Type for the Run Script Pick Action

- Once you have chosen a language, the corresponding editor can be used to write the code for the pick action. When you choose VBScript or JScript, a new script is generated in the **Script Editor**.

Run Script Parameters

<i>Parameter</i>	<i>Description</i>
Script Name	The name of the VBA macro to be executed. This name is saved in the GraphWorX file.
Script Type	Select VBA Script, VBScript or Jscript.
Create	Opens the VBA Script Wizard , which allows you to create a macro name.
Edit	Opens the corresponding script editor, which can be used to write the code for the pick action.

Action - Custom Command

The **Custom Command** pick action in the **Pick** tab of the Property Inspector can be used to run a custom function. This function can be an executable file (.exe) or a .dll file. Refer to the **Custom Configuration** section for details.

GraphWorX supports multiply custom command DLLs. In the **Prog ID** field, you can specify the Prog ID of your custom command component. If you do not provide a value, the Prog ID will be obtained from the registry by default as "IcoCustomCommand."

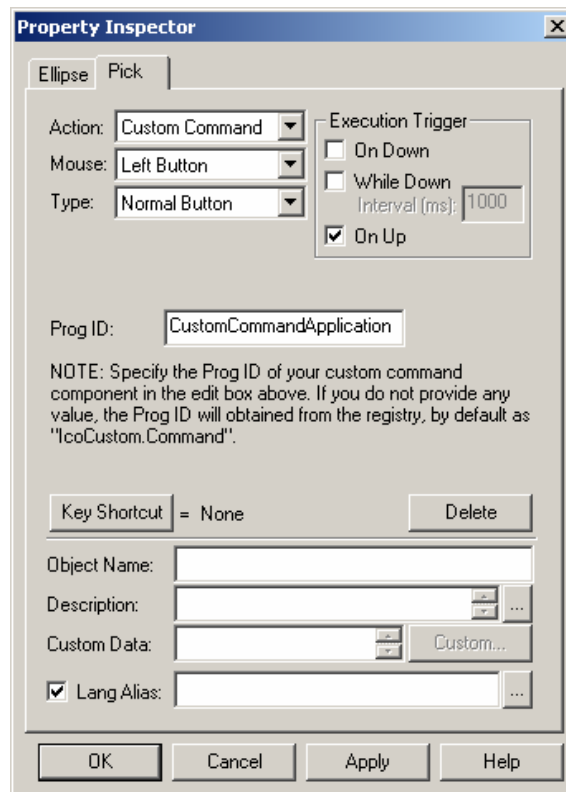


Figure 8.20. Property Inspector Dialog Box: Custom Command Action

Action - Popup Menu

The **Popup Menu** pick action in the **Pick** tab of the Property Inspector configures the font and size for popup menus opened from a pick action. You can also configure the **Popup Menu at Mouse Position**.

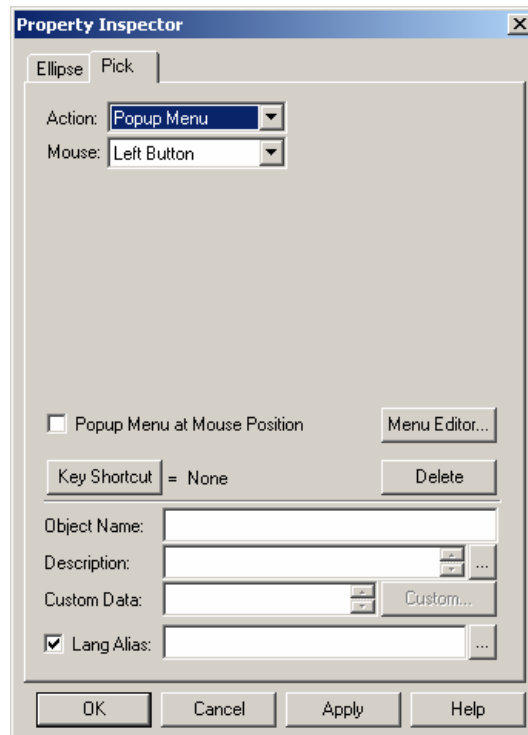


Figure 8.21. Property Inspector Dialog Box: Popup Menu Action

Click **Menu Editor** to open the **Menu Editor** dialog box, shown below, which includes fields for **Font** and **Size**. Click the **Font** button to select a font. In addition, the font face can be changed based on the loaded language by the language alias placed on the **Pick** tab. During runtime mode, the font is picked up from the language database based on the language alias and is assigned to the popup menu. This can be useful if the current/default font does not support all characters of a given language.

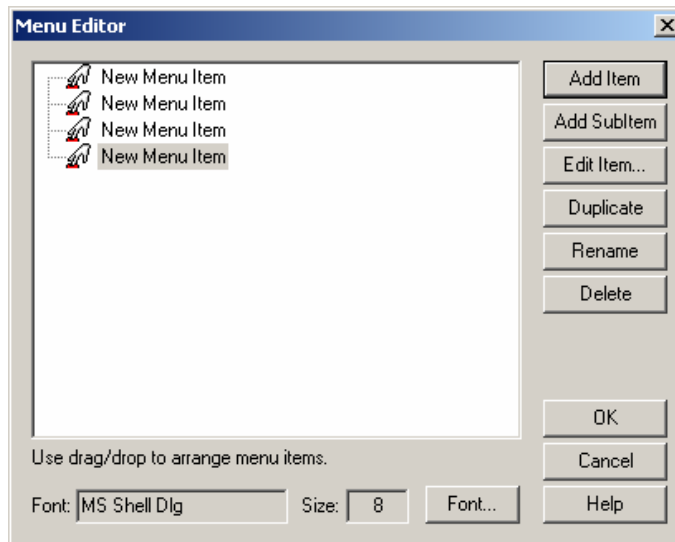


Figure 8.22. Menu Editor Dialog Box

Action - Select Language

The **Select Language** pick action in the **Pick** tab of the Property Inspector specifies languages from a pick action.

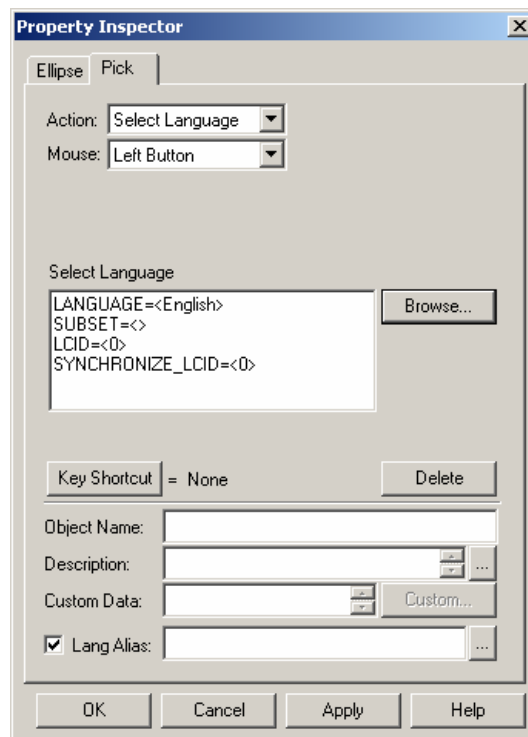


Figure 8.23. Property Inspector Dialog Box: Select Language Action

Clicking the **Browse** button opens the **Language Selector** dialog box, shown below, which enables you to choose from a list of languages in the language-switching database.

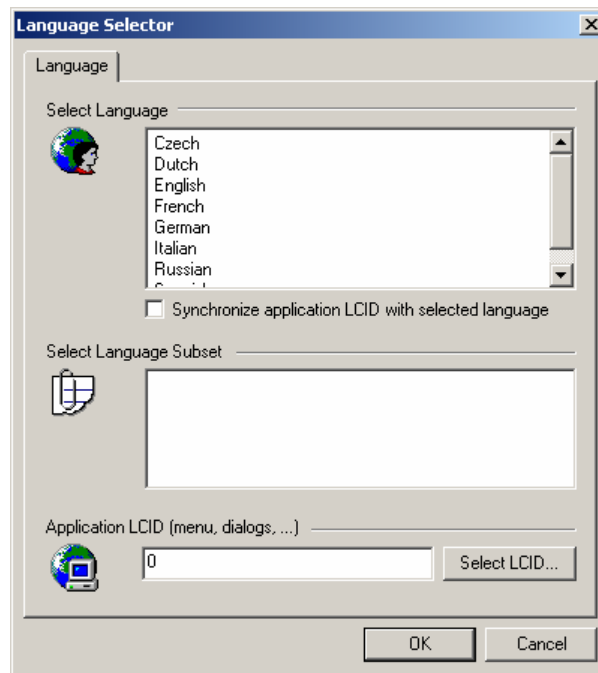


Figure 8.24. Selecting a Language

Action - Layers

You can hide, show, and toggle layers using the pick actions **Layer Hide On**, **Layer Hide Off**, and **Layer Hide Toggle** in the **Pick** tab of the **Property Inspector** dialog box, as shown in the figure below. Select the appropriate pick action, and then select a layer from the **Layer Name** drop-down list, as shown in the figure below.

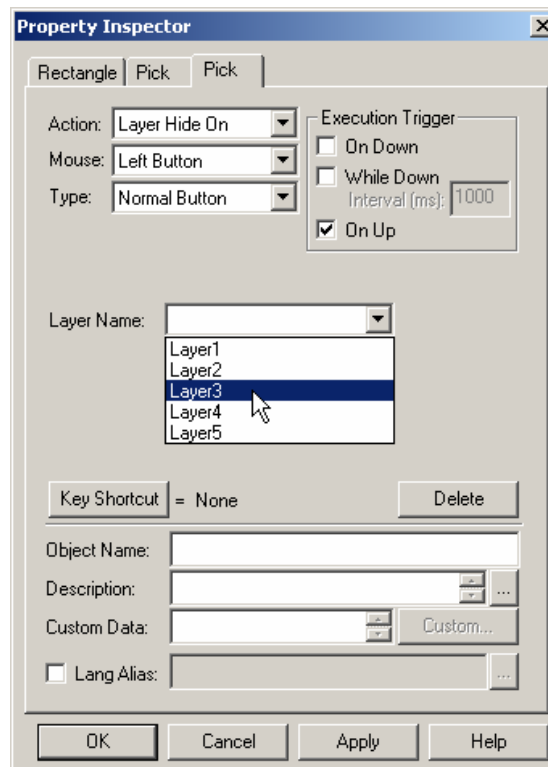


Figure 8.25. Property Inspector Dialog Box: Layers Action

Action - Select GAS Theme

The **Select GAS Theme** pick action in the **Pick** tab of the Property Inspector allows you to select global alias themes, as shown in the figure below. Click the **Browse** button to select a theme from the **Themes** dialog box, as shown in the figure below. Click **OK**.

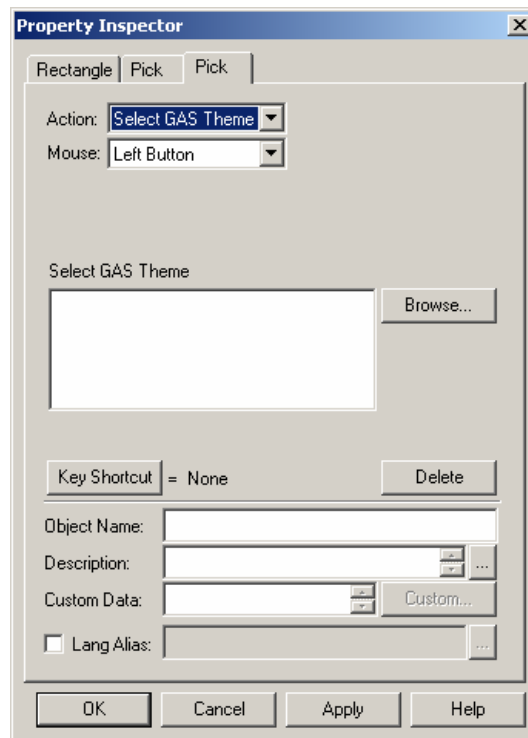


Figure 8.26. Property Inspector Dialog Box: Select Global Alias Theme Action

Note: To edit a theme's properties, select a theme and click the **Edit** button. This opens the **Theme Editor**, where you can specify theme items and the theme scope. For information about editing global alias themes and specifying the theme scope, please see the **Runtime Advanced Settings** section above.

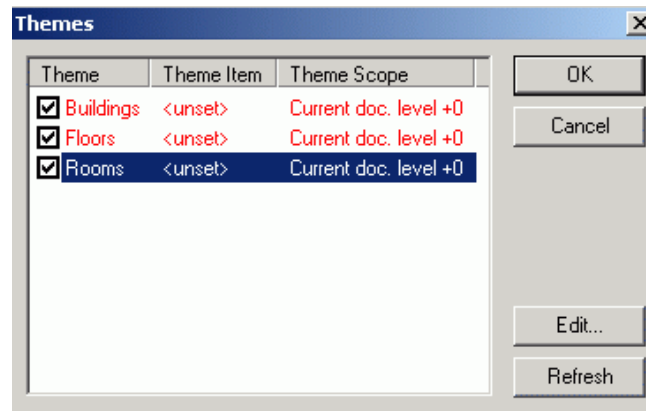


Figure 8.27. Choosing Global Alias Themes

Selector Dynamics

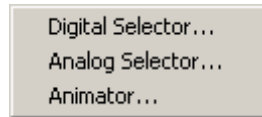


Figure 8.28. Selector Menu

Selectors display specific objects based on the value of a data connection. You can access selectors by choosing **Selectors** from the **Dynamics** menu, as shown above, or from the **Dynamics** toolbar. There are three types of selectors:

- **Digital Selector.** Displays one of a number of objects based on the state of digital signals. One signal is connected to each object. The highest priority signal (determined by position in the list) displays the corresponding object when true (or optionally false).
- **Analog Selector.** Displays one of a number of objects based on the value of the analog signal. This object displays when the signal falls within the range of values specified for the object.
- **Animator.** Selects a group of objects to display sequentially based on the state of a digital variable to which they are connected.

Note: The digital selector, analog selector, and animator can be modified after creation by using subsymbol editing. Choose the selector and either do **SHIFT+Right-click** or **Right-click - Edit Symbol** to enter subsymbol editing. Now you can add, modify, and delete the objects in the digital selector. Press the **Escape** key or double-click outside the object to close the subsymbol editing. The changes are reflected in the selector. Use the Property Inspector to connect newly added objects if necessary. An alternative way is to use the **Edit > Find** and **View** properties, or to select the inner objects from the tree.

Digital Selector



The **Digital Selector** enables you to connect individual objects to corresponding digital data points. During runtime mode, when the connected data point goes true (logical 1)(or optionally false), the connected object appears on the screen.

Only one object per connected data point can be displayed at a time. For example, if two objects are connected to the same digital data point, the object closer to the top of the connection list first appears. Generally, no objects display when all signals are false (logical 0).

To define a digital selector:

1. Select two or more objects.
2. Choose **Selectors > Digital Selector** from the **Dynamics** menu, or click the **Digital Selector** button on the **Dynamics** toolbar. The **Digital Selector** tab of the Property Inspector opens, as shown below.
3. Select the desired parameters listed in the table below.
4. Establish a data-point connection for each object using the same procedures, and then click **OK**. The objects to which the digital selector is connected are grouped together in the display.

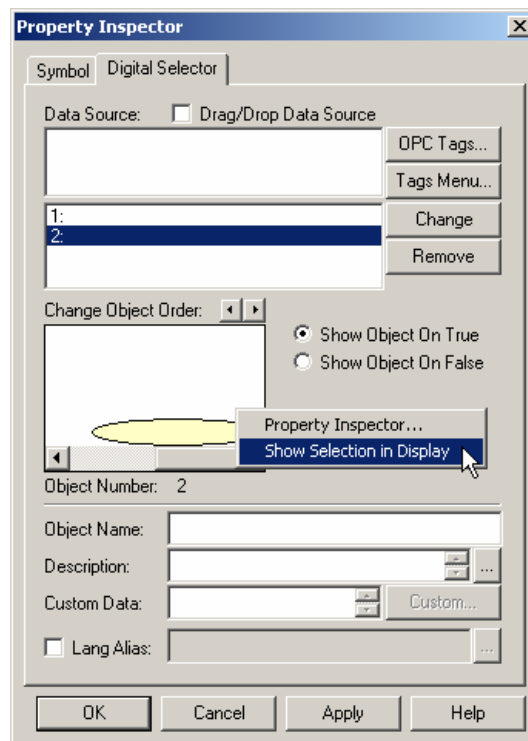


Figure 8.29. Property Inspector Dialog Box: Digital Selector Tab

Digital Selector Parameters

Parameter	Description
Data Source	Every Dynamic dialog has a data source. Data connections can be typed directly into the Data Source edit control field in the configuration dialogs of the various dynamics. Clicking the OPC Tags button opens the OPC Universal Tag Browser , from which you can select tags in the system. You can also use the options under the Tags Menu to make a data connection. These ways of making data connections for the primary value are also available for making range override data connections.
Change Button	Changes existing data connections. The currently selected item in the Data Source list box is changed to the string currently in the data source edit control.
Remove Button	Removes currently selected data connections from the data source list box.
Object Numbers	Lists the objects in the order created and their data-point connections assigning priorities to the objects.
Change Object Order	Clicking the left and right arrows to change the object order assigns new priorities for the data connections. Double-clicking one of the child objects in the Change Object Order (preview) window opens the Property Inspector for the selected child object. Right-clicking a child object (e.g. ellipse) in the Change Object Order (preview) window and selecting Property Inspector opens the Property Inspector for the selected child object. Right-clicking a child object (e.g. ellipse) in the Change Object Order (preview) window and choosing Show Selection in Display selects the current child object in the main display, as shown in the figure below. The Property Inspector can be closed and the child (which is still selected) can be modified or deleted. This function uses the sub-symbol editing feature and opens the Selector/Animator symbol for editing. It is possible to add, edit or delete all child objects within the symbol. To close the sub-symbol editing, press the Esc key or double-click outside the bounding rectangle.
Show Object on True	Shows the object when the connected data point goes true (logical 1).
Show Object on False	Shows the object when the connected data point goes false (logical 0).

Analog Selector



The **Analog Selector** dynamic defines a group of objects attached to an analog data point. During runtime mode, the value of the analog data point determines which assigned object is shown.

To define an analog selector:

1. Select two or more objects in the display.
2. Choose **Selectors > Analog Selector** from the **Dynamics** menu, or click the **Analog Selector** button on the **Dynamics** toolbar. This opens the Analog Selector tab on the Property Inspector, as shown below.
3. Select the parameters listed in the table below. Click **OK** when you are finished. The objects to which the analog selector is connected are grouped together in the display.

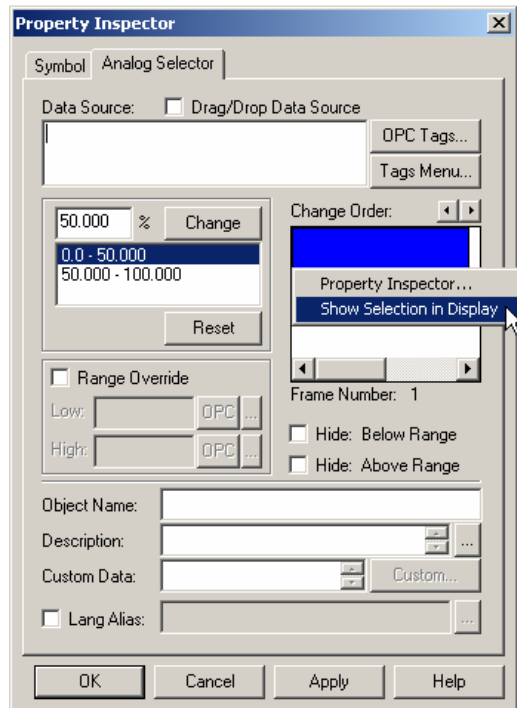


Figure 8.30. Property Inspector Dialog Box: Analog Selector Tab

Analog Selector Parameters

Parameter	Description
Data source	Every Dynamic dialog has a data source. Data connections can be typed directly into the Data Source edit control field in the configuration dialogs of the various dynamics. Clicking the OPC Tags button opens the OPC Universal Tag Browser , from which you can select tags in the system. You can also use the options under the Tags Menu to make a data connection. These ways of making data connections for the primary value are also available for making range override data connections.
Percentage	Allows objects to display when the data-point value is at a certain percentage of the full range.
Percentage List	Displays the percentage for each object.
Change Button	Changes the percentage chosen if you type a new value in the percentage field.
Reset Button	Resets the percentages chosen for the items to the default values. (The default values are an even distribution within the available range based on the number of objects selected.)

Parameter	Description
Change Order	<p>Clicking the left and right arrows to change the object order assigns new priorities for the data connections. Double-clicking one of the child objects in the Change Object Order (preview) window opens the Property Inspector for the selected child object.</p> <p>Right-clicking a child object (e.g. ellipse) in the Change Order (preview) window and selecting Property Inspector opens the Property Inspector for the selected child object.</p> <p>Right-clicking a child object (e.g. ellipse) in the Change Object Order (preview) window and choosing Show Selection in Display selects the current child object in the main display, as shown in the figure below. The Property Inspector can be closed and the child (which is still selected) can be modified or deleted. This function uses the sub-symbol editing feature and opens the Selector/Animator symbol for editing. It is possible to add, edit or delete all child objects within the symbol. To close the sub-symbol editing, press the Esc key or double-click outside the bounding rectangle.</p>
Range Override	Activates an operating range other than the default range for the data point. Specifies both a High Range and a Low Range.
Hide Below Range	Hides all the associated objects when the value of the connected data point is below the specified range.
Hide Above Range	Hides all the associated objects when the value of the connected data point is above the specified range.

Animator



The **Animator** connection enables you to select a group of objects to display sequentially based on the state of a digital variable to which they are connected. When that digital variable goes to its true state (logical 1) (or optionally false) during runtime mode, the assigned objects appear on screen in the sequence in which they are ordered. The sequence repeats until the connected process variable returns to a false state (logical 0, or optionally true).

To make an animator connection:

1. Select two or more objects.
2. Choose **Selectors > Animator** from the **Dynamics** menu, or click the **Animator** button on the **Dynamics** toolbar. The Animator tab on the Property Inspector opens, as shown below.
3. Select the parameters listed in the table below, and then click **OK**.

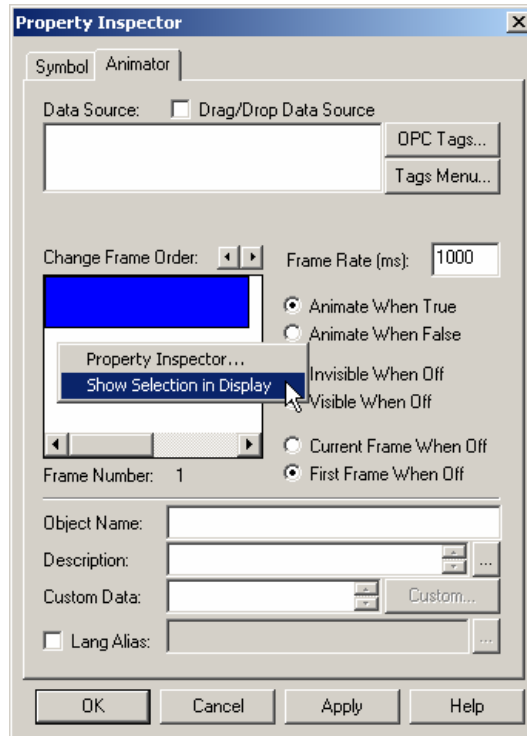


Figure 8.31. Property Inspector Dialog Box: Animator Tab

Animator Parameters

Parameters	Description
Data Source	Every Dynamic dialog has a data source. Data connections can be typed directly into the Data Source edit control field in the configuration dialogs of the various dynamics. Clicking the OPC Tags button opens the OPC Universal Tag Browser , from which you can select tags in the system. You can also use the options under the Tags Menu to make a data connection. These ways of making data connections for the primary value are also available for making range override data connections.
Change Frame Order	Clicking the left and right arrows to change the object order assigns new priorities for the data connections. Double-clicking one of the child objects in the Change Object Order (preview) window opens the Property Inspector for the selected child object. Right-clicking a child object (e.g. ellipse) in the Change Order (preview) window and selecting Property Inspector opens the Property Inspector for the selected child object. Right-clicking a child object (e.g. ellipse) in the Change Object Order (preview) window and choosing Show Selection in Display selects the current child object in the main display, as shown in the figure below. The Property Inspector can be closed and the child (which is still selected) can be modified or deleted. This function uses the sub-symbol editing feature and opens the Selector/Animator symbol for editing. It is possible to add, edit or delete all child objects within the symbol. To close the sub-symbol editing, press the Esc key or double-click outside the bounding rectangle.
Frame Number	Displays the sequence number of the currently displayed object.
Frame Rate	The rate (in milliseconds) at which the frames will change while animating.
Animate When True	Animates the object when the digital variable goes to its true state (logical 1).
Animate When False	Animates the object when the digital variable goes to its false state (logical 0).
Invisible When Off	Displays no objects when not animating.
Visible When Off	Displays the specified object when not animating.
Current Frame When Off	Displays the most recent animation frame when not animating.
First Frame When Off	Displays the first frame of the animation when not animating.

Intrinsic Dynamics

The **Intrinsic** dynamics enable you to create operator controls in display files. You can make dynamic connections to control objects that display real-time data during runtime mode, such as data entry objects, sliders, and push buttons. You can access the control functions from the **Intrinsics** submenu of the **Dynamics** menu, as shown below, or from the **Dynamics** toolbar.

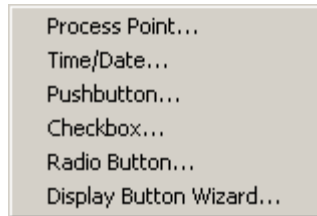


Figure 8.32. Dynamics: Intrinsics Menu

Process Points and Data Entries



The Process Point/Data Entry (PPT/DE) function creates an object used to display and enter data in an alphanumeric format. A **process point** is an object that displays the real-time value of a connected data point during runtime mode. This value is updated when the server notifies GraphWorX of a change in the data-point value.

A **data entry** behaves the same way as a process point does, but a data entry also allows you to manually enter new values to be downloaded to the system.

To create a process point/data entry (PPT/DE) object:

1. Select **Intrinsics > Process Point** from the **Dynamics** menu, or click the **Process Point** button on the **Dynamics** toolbar.
2. Click inside the work area where you want to place the process point. This opens the **Process Point/Data Entry** dialog box, as shown below.
3. Select the parameters listed in the table below.
4. Additional properties relating to the visual appearance of the PPT/DE can be made by selecting the **Text** tab. The text properties for Process Points are similar to those used to edit static text objects.
5. Click **OK**. A box with question marks representing the PPT/DE format selected appears in the work area.

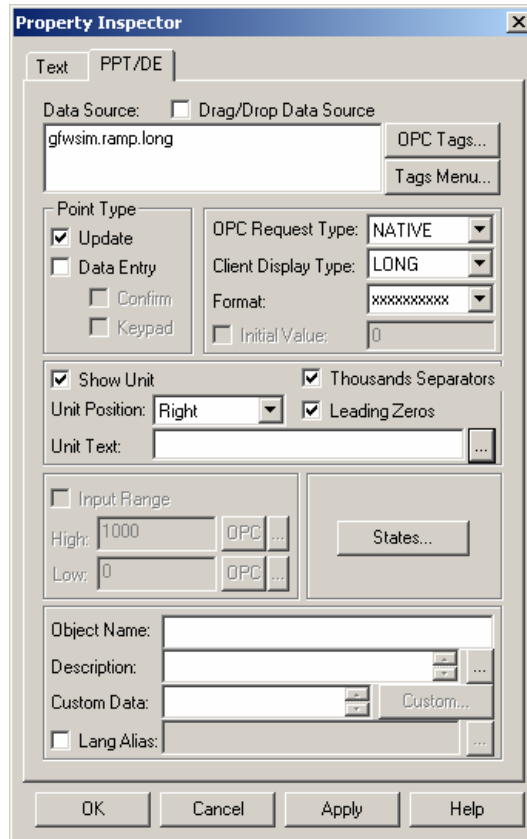


Figure 8.33. Property Inspector Dialog Box: Process Point Tab

Process Point/Data Entry Parameters

Parameter	Description
Data Source	Every Dynamic dialog has a data source. Data connections can be typed directly into the Data Source edit control field in the configuration dialogs of the various dynamics. Clicking the OPC Tags button opens the OPC Universal Tag Browser , from which you can select tags in the system. You can also use the options under the Tags Menu to make a data connection. These ways of making data connections for the primary value are also available for making range override data connections.
Update	Defines a point type that the system updates in real-time.
Data Entry	Changes the value of the connected data point in runtime mode.
Keypad	Displays a keyboard or number pad to allow a value to be entered when the process point is clicked during runtime mode. You can set the size of the keypads using the Runtime tab of the Application Preferences dialog box.
OPC Request Type	The OPC Request Type is used in communicating OPC data between GraphWorX (an OPC client) and an OPC server, and between the OPC server and the I/O device(s) to which the OPC server is connected.
Client Display Type	The Client Display Type specifies how OPC data from an OPC server are shown in the GraphWorX display.
Format	Specifies the number of decimal places for the connected data point in runtime.
Float	Floating decimal point. Offers many default formats, such as "xxx.xx," in a drop-down list for analog (floating point) process points. You can also define a scientific notation format using the predefined formats, or you can create a user-defined format. An example of scientific notation formats is the "x.xxe" format.
Double	Double precision. Offers the same formats as Float.
Bool	Boolean. Defines two-state display formats. Offers default formats, such as 0/1, Off/On, Yes/No, False/True, Auto/Manual, Enable/Disable, and Stop/Run.
Byte	8 bits. Offers the same formats as Short and Long.

Parameter	Description
Word	32 bits. Offers the same formats as Short and Long. "Word" is an unsigned 16-bit value.
DWord	64 bits (double word). Offers the same formats as Short and Long. "Double word" is an unsigned long value (32 bits).
Char	Character. Offers the same formats as Short and Long. "Char" is a signed 8-bit value.
Short	Short integer. Offers 10 default formats for integers, such as xxxx.
Long	Long integer Offers the same formats as Short.
String	Any text string. Defines String text display formats, 10, 20, 30, 40, 50, 60, 70, and 80, as the number of characters. For String process points, you must define the number of text characters before you place this object in the display. Eight predefined string lengths are provided in the menu. You can use the predefined format or create your own.
Hexadecimal (Client Display only)	Hexadecimal constant. e.g. $0x20A = 2 * (16^2) + 0 * (16^1) + 10 * (16^0) = 2*256 + 0*16 + 10 * 1 = 512 + 0 + 10 = 522$
Octal (Client Display only)	Octal constant. e.g. $0t36 = 3 * (7^1) + 6 * (7^0) = 3*7 + 6*1 = 21 + 6 = 27$
Binary (Client Display only)	Binary constant. e.g. $0b110 = 1 * (2^2) + 1 * (2^1) + 0 * (2^0) = 1 * 4 + 1 * 2 + 0 * 1 = 4+2+0 = 6$
Native (OPC Request only)	Unspecified data type.
Initial Value	Defines the initial (startup) value to which the data-entry object is initialized when the display is launched in runtime mode.
Show Unit	Check this check box to display a unit of measurement following (or preceding) the data value during runtime. The unit position can be left or right. The text can be anything (e.g.: "meters"). The Unit Text can also be a global alias or a language alias.
Thousands Separators	Check this check box to show thousands separators in runtime. The thousand-separator character (e.g. commas) depends on the regional setting. For example, in runtime the number "13200" (13 thousand two hundred) will appear as "13,200" in the United States and as "13.200" in Europe.
Leading Zeros	Check this check box to show leading zeros to the left of the data value in runtime; there will be as many leading zeros as required to reach the specified format length (number of decimal places). For example, if the format is "xxxx" and the value is "23," then the process point will show "0023."
Input Range	Specifies a low and high range in the boxes provided. This range applies to data entries only. In runtime mode, GraphWorX will not allow you to enter a value outside this range.
States	Opens the State Field Configuration dialog box, which allows you to enter the meaning of a binary value. See the State Fields section for more information.

State Fields

Clicking the **States** button in the **PPT/DE** tab of the Property Inspector opens the **State Field Configuration** dialog box, shown below, which allows you to enter the meaning of a binary value. **State fields** refer to the state of a variable. They are usually a binary value, which is attached to a string, which in turn has some meaning.

Note: The **Find / Replace** feature on the **Edit** menu is now supported for state fields. In the Find dialog box, select **Text Label** from the **Type** drop-down list, and then enter your search parameters.

Note: The state field in GraphWorX is only able to display a single line of text.

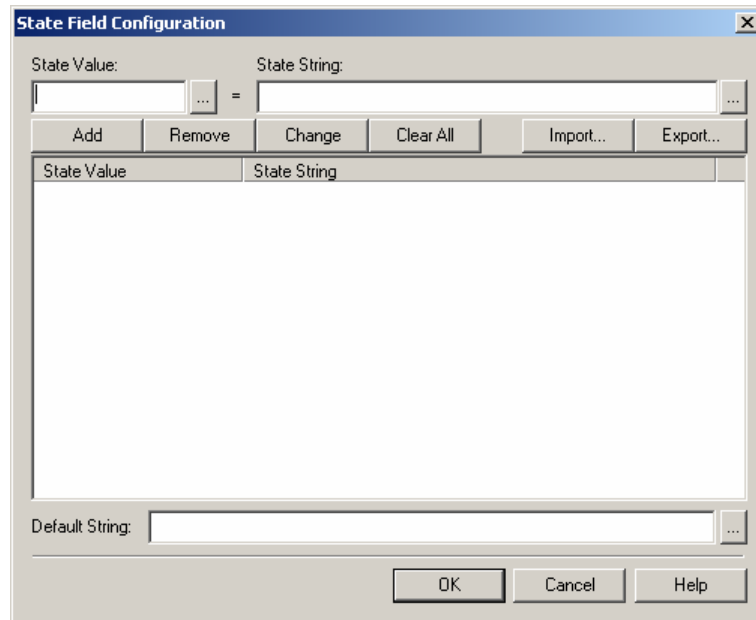


Figure 8.34. State Field Configuration Dialog Box

The configuration of state fields is very similar to that of aliases. You can enter a state value or simply select a state that already exists, enter or change the state string, and then click the proper action button (**Add**, **Remove**, **Clear** or **Clear All**). The **Import** and **Export** buttons open the **Open** and **Save As** dialogs, respectively. The **Default String** refers to the string that will appear when the state does not have a preconfigured string attached to it.

To select an alias, click the ... button to the right of the **State Value**, **State String**, and **Default String** text boxes and select either **Global Aliases** or **Language Aliases** from the pop-up menu, as shown in the figure below. This opens the respective alias browsers, allowing you to choose from a list of available aliases.

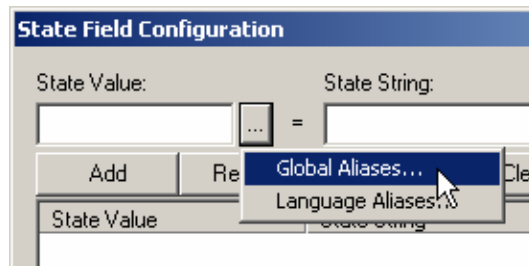


Figure 8.35. Selecting an Alias To Use for State Fields

If the process point is set to **Data Entry**, a drop-down menu of all possible states will be available during runtime, as shown in the example below.

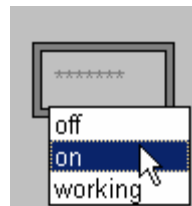


Figure 8.36. Data Entry Process Point In Runtime Mode

Time/Date



Different options are provided for the **Time** and **Date** functionality under the **Dynamics** menu of GraphWorX.

To define **Time/Date**:

1. Select **Intrinsics > Time/Date** from the **Dynamics** menu, or click the **Time/Date** button on the **Dynamics** toolbar.
2. Click the work area where you want the Time/Date to appear.
3. This opens the **Time/Date** tab on the Property Inspector, as shown below.
4. Select a time and date format. Scroll through the various predefined formats in the format list box.
5. You can configure additional properties relating to the visual appearance of the Time/Date object using the **Text** tab, which is similar to that used to edit static text objects.
6. **Click OK**. The Time/Date object appears in your display.

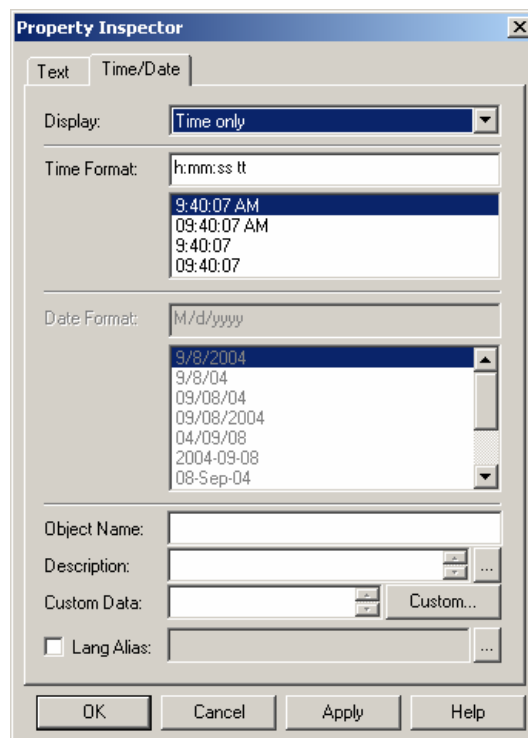


Figure 8.37. Property Inspector Dialog Box: Time/Date Tab

Push Button



The **Pushbutton** function creates a push button in a GraphWorX display. The push button object behaves almost exactly like the pick dynamic described earlier in this section. In addition to pick dynamic functionality, push buttons animate the up/down state in runtime mode.

To create a push button:

1. Select **Intrinsics > Push Button** from the **Dynamics** menu, or click the **Button** button on the **Dynamics** toolbar.
2. Click the work area where you want the pushbutton to appear.
3. This opens the Property Inspector for pushbuttons, as shown below. The Property Inspector includes tabs for **Pick** (the same tab as the pick dynamic) and **Button** (the same tab as the static text object).

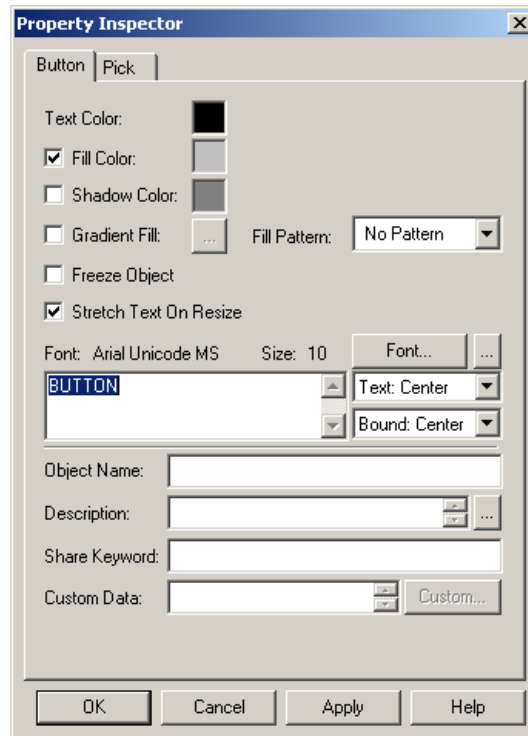


Figure 8.38. Property Inspector Dialog Box: Button Tab

4. Configure the properties for the push button, and then click **OK** to save these changes. The push button appears in the display, as shown below.



Check Box



The **Check Box** function allows you to create a check box in the same way you can create a push button. Check boxes differ slightly from normal push buttons in both appearance and behavior. GraphWorX check boxes look and act like standard Windows check box controls. When selected in runtime mode, the check box remains selected until it is clicked again.

To create a check box:

1. Select **Intrinsics > Check Box** from the **Dynamics** menu, or click the **Check Box** button on the **Dynamics** toolbar.
2. Click the work area where you want the check box to appear.
3. This opens the Property Inspector for check boxes. The Property Inspector includes tabs for **Pick** (the same tab as the pick dynamic) and **Check Box** (the same tab as the static text object).

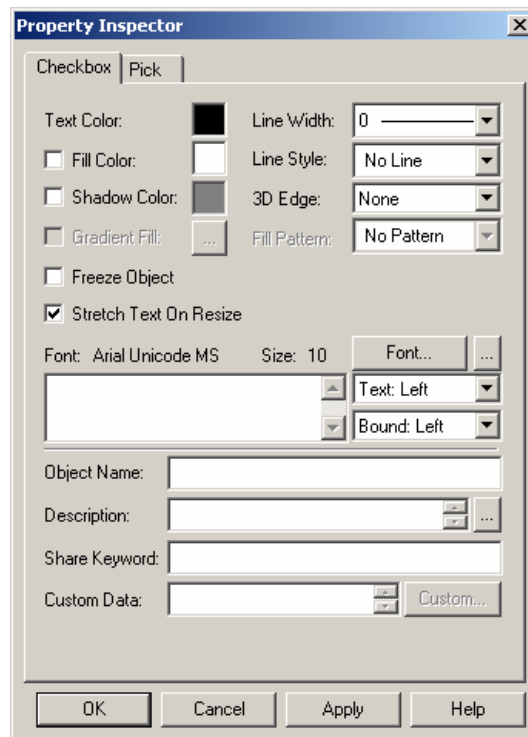


Figure 8.39. Property Inspector Dialog Box: Check Box Tab

4. Configure the properties for the check box, and then click **OK** to save these changes. The check box appears in the display.

Tracking for Check Boxes

If the **Value Tracking** option is selected on the **Pick** tab of the Property Inspector, as shown in the figure below, then the check box that you are creating in your display will be checked or unchecked in runtime mode based on the connected data source tag value.

For example, create several check boxes in your display and connect them to the same tag (it can be a local variable). If the **Value Tracking** option is selected, a change made to one check box will be reflected by the other check boxes as well. Without the tracking option, the other check boxes do not change automatically.

Note: A similar example can also be used for the radio button.

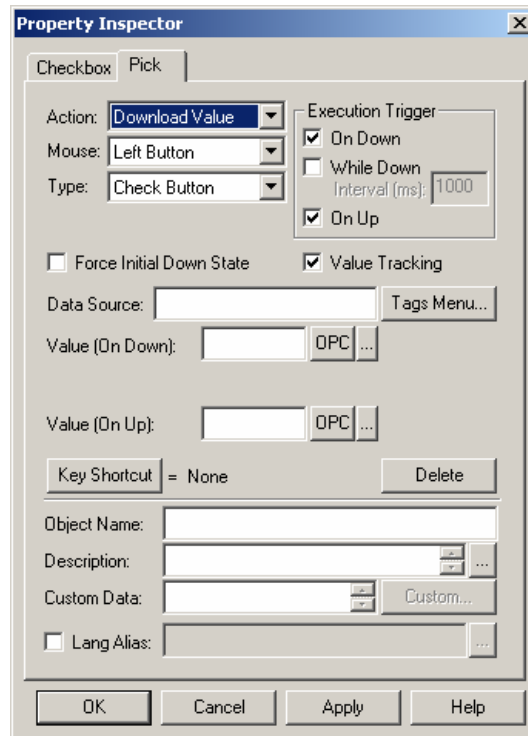


Figure 8.40. Value Tracking for Check Boxes

Radio Button



The **Radio Button** function allows you to create a radio button in the same way you can create a push button or a check box. Radio buttons differ slightly from normal push buttons in both appearance and behavior. GraphWorX radio buttons look and act like standard Windows radio buttons controls. When pressed in runtime mode, the radio button becomes selected and automatically deselects all other radio buttons that belong to the same group.

To create a radio button:

1. Select **Intrinsics > Radio Button** from the **Dynamics** menu, or click the **Radio Button** button on the **Dynamics** toolbar.
2. Click the work area where you want the radio button to appear.
3. This opens the Property Inspector for radio buttons. The Property Inspector includes tabs for **Pick** (the same tab as the pick dynamic) and **Radio Button** (the same tab as the static text object).
4. Configure the properties for the radio button, and then click **OK** to save these changes. The radio button appears in the display.

Tracking for Radio Buttons

If the **Value Tracking** option is selected on the **Pick** tab of the Property Inspector, as shown in the figure below, then the radio button that you are creating in your display will be selected or deselected in runtime mode based on the connected data source tag value.

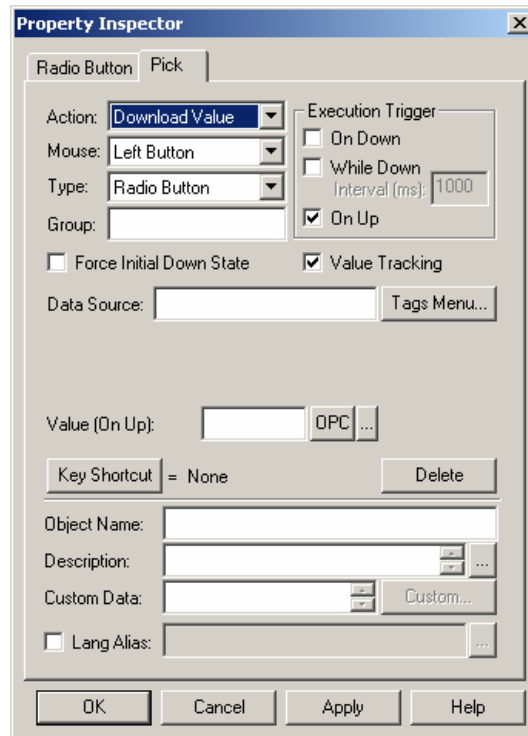


Figure 8.41. Value Tracking for Radio Buttons

Display Button Wizard

GraphWorX contains a feature called a **display button**. A display button, once configured through a wizard, can do one of four things:

- Load a new display.
- Open a display in an embedded GraphWorX window.
- Display a popup window.
- Serve as a drag-and-drop object that can be placed in a GraphWorX ActiveX container to load a display.

This feature reduces clutter in a display because embedded windows and popup windows are shown only when needed by the push of a button.

To configure a display button, select **Intrinsics > Display Button Wizard** from the **Dynamics** menu, or click the **Display Button** button on the **Dynamics** toolbar. This opens the **Display Button Wizard**, shown in the figure below.

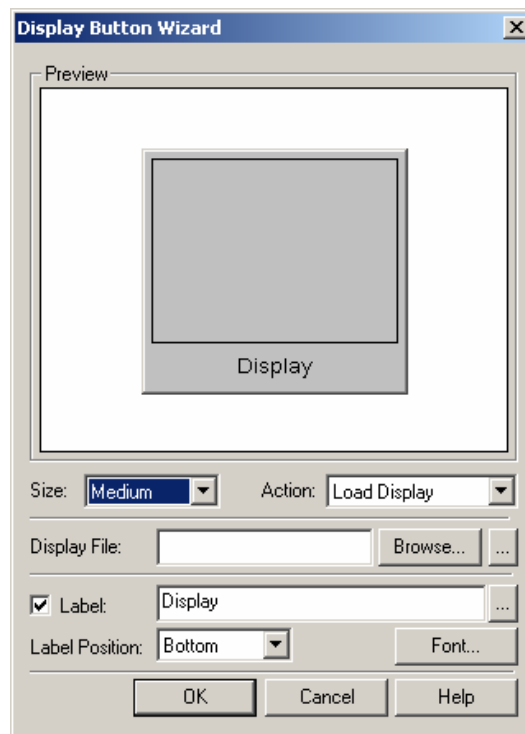


Figure 8.42. Display Button Wizard

The **Display Button Wizard** is a very simple, single-dialog wizard. The **Preview** field shows what the surface of the button will look like once it is configured. The options available for customizing both the appearance and functionality of the button are outlined in the following sections.

Note: Once the **Display Button Wizard** settings are changed, they will default to these settings for the remainder of the current editing session.

To select an alias, click the ... button to the right of the **Display File** and **Label** text boxes and select either **Global Aliases** or **Language Aliases** from the pop-up menu, as shown in the figure below. This opens the respective alias browsers, allowing you to choose from a list of available aliases.

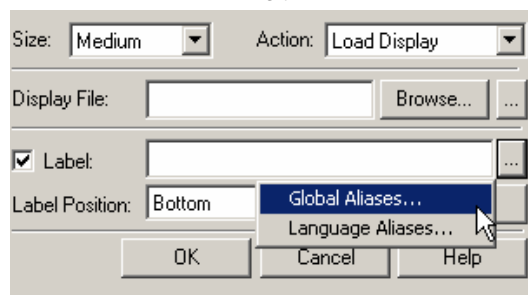


Figure 8.43. Selecting an Alias To Use for Display Buttons

Display Button Wizard - Size

The **Size** field in the **Display Button Wizard** allows you to select the size of the button from the drop-down list. The button in the **Preview** window resizes to match the item selected to show you exactly what size the button will be.

Display Button Wizard - Display File

In the **Display File** field in the **Display Button Wizard**, enter the name of the file that the display button will load. You can use the **Browse** button to search for and select a file. Once a file is selected, the button will show a shrunken image of the display it is configured to load.

To select an alias, click the ... button to the right of the **Display File** text box and select either **Global Aliases** or **Language Aliases** from the pop-up menu, as shown in the figure below. This opens the respective alias browsers, allowing you to choose from a list of available aliases.

Display Button Wizard - Action

This combo box determines how the button will behave in runtime mode. Each of the four available actions is listed below and uses the display entered in the **Display File** field.

Load display: Closes the current display and opens the new display.

Embedded window: Opens the new display in an embedded GraphWorX window, leaving the original display open.

Popup window: Opens a popup window that, unlike the embedded window, is not restricted to the inside of the GraphWorX display and can be maximized or minimized by the user.

Drag/drop load: When this option is chosen, the button no longer behaves like a standard button. Rather it behaves like a drag/drop object. If there are existing GraphWorX containers in the display, the button can then be dragged and dropped into the container, causing the specified display to load.

Display Button Wizard - Label

In the **Label** field of the **Display Button Wizard**, you can specify which label is to appear on the display button. If you do not wish to have a label on the button, uncheck the check box next to **Label**.

To select an alias, click the ... button to the right of the **Label** text box and select either **Global Aliases** or **Language Aliases** from the pop-up menu, as shown in the figure below. This opens the respective alias browsers, allowing you to choose from a list of available aliases.

Clicking the **Font** button to choose the font type. The appropriate attributes from the standard font window can change the label font. You can put the label near the top or near the bottom of the button by selecting either **Top** or **Bottom** in the combo box next to **Label Position**.

Once the wizard is complete, the display button is ready to be used in runtime mode. Upon execution, the button will perform according to the way it was configured in the wizard. The figure below shows three different display buttons. Each one launches a simple display called "Ellipse.gdf," which is simply an ellipse configured to rotate through the use of a simulated tag. One display button is configured to launch a popup window. The second launches an embedded window. The third is a Drag/Drop button located next to a GraphWorX ActiveX container.

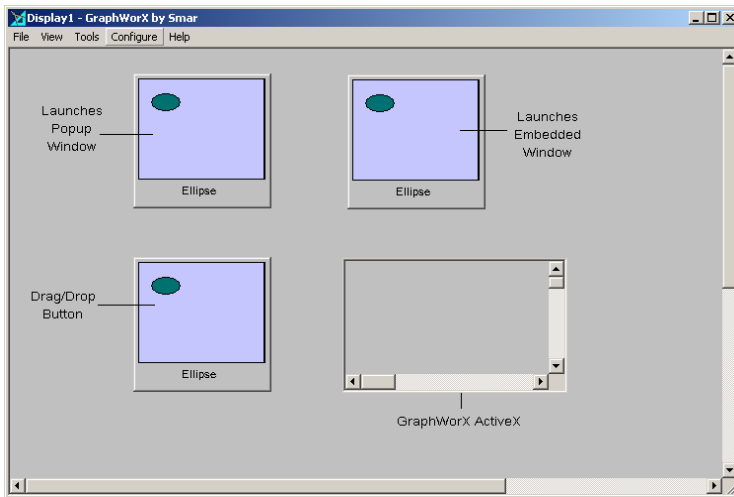


Figure 8.44. Demonstration of Different Display Buttons in Runtime

Three different types of runtime windows, each with its own independently animated ellipse, all come into existence in runtime through the use of display buttons. At the top left of the figure is a demonstration of the effect of a display button used to launch a popup window. At the top right, an embedded window display button is shown. As you can see, the popup window can be minimized and maximized, while the embedded window cannot. At the bottom of the figure is a Drag/Drop display button next to a GraphWorX container that was, in configuration mode, an empty container. Dragging that button and dropping it into the container caused the ellipse display to load in the container. Had there already been a display in the container, it would have been replaced by the new display. In a more practical setting, multiple displays can be called by one master display through the use of display buttons. Entire displays get encapsulated in small, user-defined buttons ready to be activated at the user's request.

Customizing

Custom Configuration

Every tab in the GraphWorX Property Inspector has a **Custom** button located next to the **Custom Data** edit field. This button can be tied to a custom configuration utility. The custom configuration utility has to be programmed by the user.

Clicking this button will create an Automation object that has a certain program ID (ProgID). By default the ProgID is **lcoCustom.Configure**. This ProgID can be overridden by setting an alternative ProgID in the Windows registry key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\ICONICS\Gwx32\OEM Information\Custom
Configure ProgID = "YourProgIDHere"
```

You will probably want this Automation object to be exposed from a DLL. GraphWorX then uses the dispatch interface of this object to call the method:

```
boolean CustomConfigure(long hWndParent, IDispatch* ObjectOne, BSTR
ObjectOneType, IDispatch* ObjectTwo, BSTR ObjectTwoType);
```

Explanation:

hWndParent - the window handle of GraphWorX property inspector. You will probably want to popup a modal dialog as your custom configuration UI. Use this window handle as the parent of your dialog.

ObjectOne - dispatch pointer of the GraphWorX object associated with the Gwx property page currently shown (i.e. if the "pick" page is currently active, ObjectOne is the "GwxPick" object for that page).

ObjectOneType - String name of the object type for ObjectOne (for example, "GwxPick"). You will probably want to check the object type to be sure that you support configuration for this object type.

ObjectTwo - dispatch pointer of the GraphWorX Display to which ObjectOne belongs.

ObjectTwoType - String name of the object type for ObjectTwo (i.e. "GwxDisplay").

Return value is TRUE if you made changes to the objects passed to this method. FALSE if no changes were made (i.e. Cancel). DO NOT Release() the dispatch pointers passed to this method!!!

After the method returns, GraphWorX releases the dispatch pointer for the "lcoCustom.Configure" automation object.

Below is some sample code for a possible implementation of the **CustomConfigure** method:

```
BOOL CAutoConnect::CustomConfigure(long hWndParent, LPDISPATCH
ObjectOne, LPCTSTR ObjectOneType, LPDISPATCH ObjectTwo, LPCTSTR
ObjectTwoType)
{
    CString obj1Type(ObjectOneType);
    if (obj1Type == _T("GwxPick"))
    {
        CString strCustomData;
        //Run you configuration dialog here...
        if (returnValOfConfigDialog)
        {
            COleDispatchDriver obj1(ObjectOne, FALSE);
            //set user custom data
            obj1.SetProperty(0x10001, VT_BSTR,
(LPCTSTR)strRCustomData);
            //set pick action = Custom Command
            obj1.SetProperty(0x1, VT_I4, (long)8);
            return TRUE;
        }
    }
}
```

```
else
{
    AfxMessageBox(_T("Custom Configuration for this object type
    is not supported."));
}
return FALSE;
}
```

Data and Timer Threads

The **Data Thread** and **Timer Thread** can be customized by two registry settings. If there is a need to increase the thread priority of GraphWorX, it is possible to configure these registry keys in the registry:

```
HKEY_LOCAL_MACHINE\SOFTWARE\ICONICS\Gwx32\Runtime
Settings\DataThread\Priority
```

```
HKEY_LOCAL_MACHINE\SOFTWARE\ICONICS\Gwx32\Runtime
Settings\TimerThread\Priority
```

The values are strings, and the following strings are valid:

-15	THREAD_PRIORITY_IDLE
-2	THREAD_PRIORITY_LOWEST
-1	THREAD_PRIORITY_BELOW_NORMAL
0	THREAD_PRIORITY_NORMAL [default]
1	THREAD_PRIORITY_ABOVE_NORMAL
2	THREAD_PRIORITY_HIGHEST
15	THREAD_PRIORITY_TIME_CRITICAL

Custom Command Execution

This is one of the options for the PICK dynamic action in GraphWorX. You can run custom functions/applications using the Custom Command option. The custom application has to be specified by the user. The application/function is either an executable (.EXE) or a DLL.

The first time the user clicks on a pick action that is a **Custom Command**, GraphWorX creates an Automation object that has a certain program ID (ProgID).

By default the ProgID is: **IcoCustom.Command**. This ProgID can be overridden by setting an alternative ProgID in the registry key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\ICONICS\Gwx32\OEM Information\Custom
Command ProgID = "YourProgIDHere"
```

You will probably want this Automation object to be exposed from an EXE. The EXE should allow you to create multiple instances of this automation object without launching multiple instances of the application.

GraphWorX then uses the dispatch interface of this object to call the method:

```
boolean CustomCommand(long hWndParent, IDispatch* ObjectOne, BSTR
ObjectOneType, IDispatch* ObjectTwo, BSTR ObjectTwoType);
```

The parameters are the same as CustomConfigure() described in the previous section. **DO NOT Release() the dispatch pointers passed to this method!!!**

After the method returns, GraphWorX does NOT release the dispatch pointer for the "IcoCustom.Command" automation object. GraphWorX keeps the dispatch pointer for the next time a Custom Command is executed so that GraphWorX does not have to create the object again (so execution is faster). This dispatch pointer is not released until this instance of GraphWorX is closed.

Section 9

Tools

Tools Menu

The **Tools** menu allows the user to access many tools that are used in the manipulation of GraphWorX data. The following is a list of options available from this list:

- Macros (Create, Edit, Step, Run, Clean Unused VBA Modules, and open the Visual Basic Editor). (For more information on the Visual Basic Editor, please refer to the Visual Basic for Applications documentation.)
- Functions keys
- Set Working Directory
- Security Configuration (For more information, please refer to the Security Configurator documentation.)
- Local Alias File Editor
- Global Aliasing Configuration (For more information, please refer to the Global Aliasing documentation.)
- Language Aliasing Configuration (For more information, please refer to the Language Aliasing documentation.)
- 16 bit to 32 bit Translator (For more information, please refer to the GraphWorX Translation Utility documentation.)
- Publish to HTML
- Windows CE (Configure for Windows CE and Windows CE Preferences)

Function Keys

The **Function Keys** feature available on the **Tools** menu provides simple function keys management and a hierarchical functionality. Selecting **Function Keys** from the **Tools** menu opens the **Function Keys Script Editor**, shown below, which allows you to add, edit, and remove triggers. A **trigger** is a combination of a shortcut key and a VBA script name, which together allow the user to open a VBA script when the shortcut key is pressed.

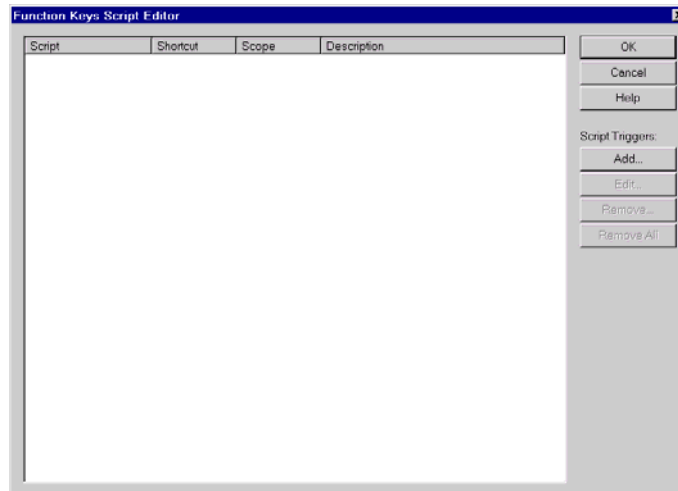


Figure 9.1. Function Keys Script Editor

Clicking the **Add** button or the **Edit** button opens the **Function Key Properties** dialog box, shown below, which enables you to specify the shortcut and the script.

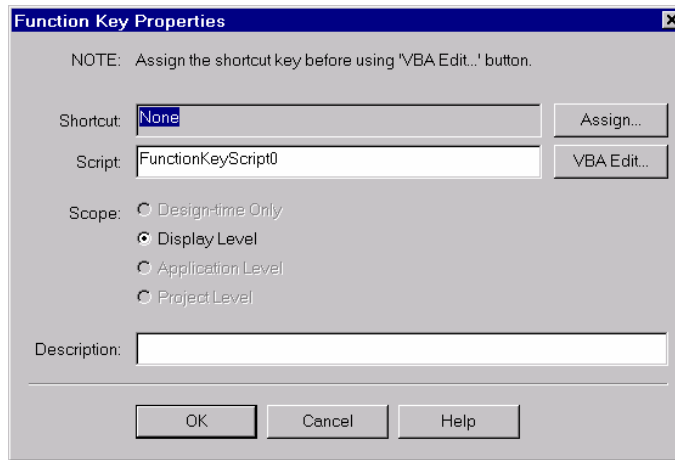


Figure 9.2. Function Key Properties

You can either type the script name or click the **VBA Edit** button, which creates a script or jumps to the script if the script already exists. You can now type the script body in the VBA editor. It is recommended that you debug the script and verify that it compiles without problems by clicking **Debug** and **Compile Project** in the VBA Editor.

The **Scope** field specifies the scope of the script. Currently only the **Display Level** script scope is available. Display level function keys are the first step of the function keys hierarchy. The other scope options are disabled.

The **Description** field is simply a text field that appears in the last column of the **Function Keys Script Editor**.

Set Working Directory

Selecting **Set Working Directory** from the **Tools** menu opens the **Set Working Directory** dialog box, shown below, which enables you to configure a custom directory in which all application configuration files will be stored and retrieved.

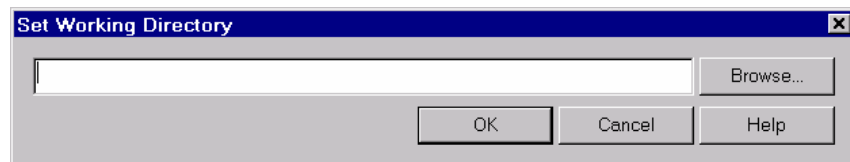


Figure 9.3. Set Working Directory Dialog Box

Click **Browse** to select the directory from the **Browse for Folder** dialog, as shown in the figure below.

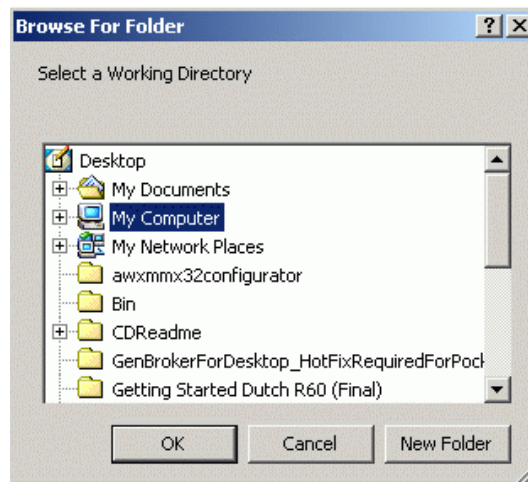


Figure 9.4. Selecting a Working Directory

Security

GraphWorX has an interface with the Security Server and currently supports the security actions shown in the dialog box below.

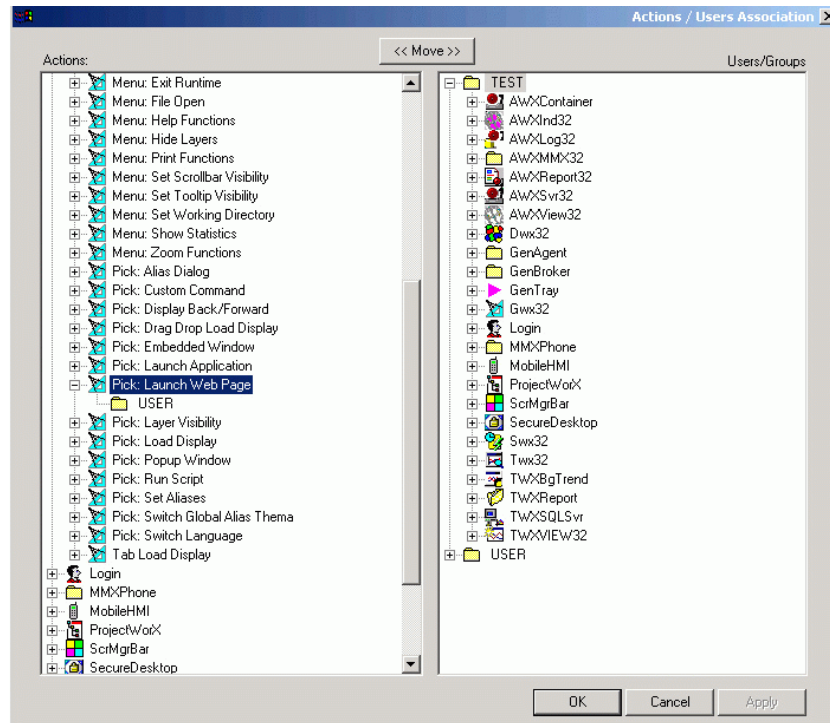


Figure 9.5. GraphWorX Security Actions

For more information about security, please refer to the Security Configurator help documentation.

Local Alias File Editor

The **Alias File Editor** command on the **Tools** menu opens the **Alias File Editor** dialog box, shown below, which allows you **Add**, **Remove**, and **Change** aliases. Click on the **Aliases** button to show aliases for the display. You can also choose to **Import** or **Export** an alias. Under the **Alias Definition** field, you can define aliases using the **Tags Menu** button, which allows you to select tags, expressions, local variables, simulation variables, and global aliases.

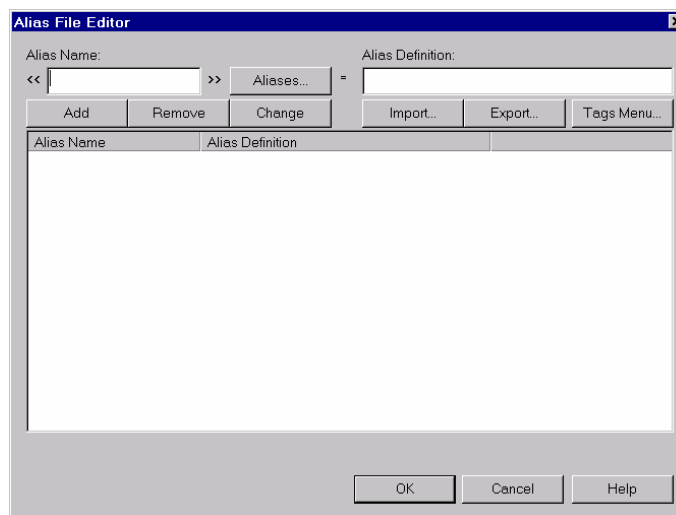


Figure 9.6. Local Alias File Editor Dialog Box

Publishing to HTML

The **Web Publishing Wizard** enables you to "export" your GraphWorX (.gdf) display over to HTML files to a Web server (LAN or Internet). In publishing the display to a Web server, WebHMI uses HTML to reference the .gdf files in an Internet-enabled format. Once a GraphWorX display is "exported" to an HTML file and then published to a Web server, client machines can browse it through an Internet browser, such as Microsoft Internet Explorer or Netscape Navigator. Each GraphWorX display can be viewed as a Web page.

Note: Netscape Navigator generally does not support ActiveX technology. Smar has solved this problem by supplementing Netscape with a plug-in. Acting as a bridge, this plug-in allows for an ActiveX component to be put into the special Netscape module.

Basic Steps in Publishing GraphWorX Displays

Publish a GraphWorX display to HTML involves the following basic steps:

1. Create a GraphWorX display (.gdf) file on a developer workstation with GraphWorX installed.
2. Use the Web Publishing Wizard to generate an HTML file and publish the file to a Web server (URL) address
3. On the Web server PC (with ProcessView installed), start GenBroker Server from ProcessView Tray.
4. On a client PC, open a Web browser, such as Microsoft Internet Explorer, and browse to the URL address of the HTML file. The client PC downloads the HTML file from the Web server. The Web server delivers all the necessary components (e.g. ActiveX controls) to the client PC's Web browser, which then runs the GraphWorX display directly within the browser. This way you can view the GraphWorX display in real time as a Web page.

The source code of the published HTML file contains references to the required "plug-ins" that are needed to deliver the GraphWorX Viewer ActiveX control to the client. The GraphWorX Viewer ActiveX is needed to run the .gdf file over the network. The GraphWorX display itself is not really "converted" into HTML. Instead the display (.gdf) file is referenced in the HTML code by the ActiveX plug-in.

To expand on step 2 above, the following Microsoft Internet Explorer example shows the portion of the generated HTML source code that references the .gdf file. The **GWXview32.cab** file is referenced in the **codeBase** field to deliver the "plug-in" for the GraphWorX Viewer ActiveX. This in turn opens up the .gdf display. In the example below, the .gdf display file is called **Building1.gdf**, as shown below in the **DisplayName** parameter field.

```
<OBJECT classid=clsid:98A5DDE3-563B-11CF-A343-487C03C10000
  codeBase="http://www.myserver.com/webhmi/cabs/GWXview32.cab"
  id=GWXview321 style="HEIGHT: 420px; WIDTH: 620px" height=420
  width=620>
  <param name="_Version" value="65537">
  <param name="_ExtentX" value="16404">
  <param name="_ExtentY" value="11113">
  <param name="_StockProps" value="160">
  <param name="BorderStyle" value="1">
  <param name="Appearance" value="1">
  <param name="DisplayName"
  value="http://www.myserver.com/webhmi/Building1.gdf">
  <param name="UseAmbientBackColor" value="0">
  <param name="AutoStartRuntime" value="-1">
  <param name="OverrideScrollbarSettings" value="1">
  <param name="VerticalScrollbar" value="0">
  <param name="HorizontalScrollbar" value="0">
</OBJECT>
```

Exporting a .gdf file to an HTML file not only references the GWXview32.cab file, as shown in the example above, but will also references any other "plug-in" .cab files required for other ActiveX components that may be embedded within the .gdf file (e.g., TrendWorX or AlarmWorX Viewer ActiveX controls).

However, before a .gdf files can communicate with live OPC data, the **IcoSetServer.cab** file“plug-in,” which is necessary for security and licensing, as well as a **GenBroker configuration (.gbc or .gbx) file**, which contains network configuration settings for OPC communications, must also be delivered to the client PC. The IcoSetServer.cab file and the .gbc or .gbx file are referenced in the HTML source code. This way the security and licensing information are available whenever a component (such as an Alarm Viewer ActiveX) is downloaded. The sample HTML source code below shows how these files are referenced in the code.

```
<object id="SetServer2" classid="clsid:57802C16-9A15-11D4-B2A8-0090272E599B"
codeBase=http://www.myserver.com/WebHMI/cabs/IcoSetServer.cab
height=28 width=17>
  <PARAM NAME="CFgName"
VALUE="http://www.myserver.com/WebHMI/Samples/Default.gbc">
</object>
```

The Web Publishing Wizard in GraphWorX takes care of all the necessary HTML code references automatically. Microsoft Internet Explorer uses .cab files, whereas Netscape Navigator uses .dpl files.

Note: If you are using multiple frames for your WebHMI pages (e.g. a main "navigation" frame for browsing between pages, and a "content" frame that contains the body of the pages), make sure that the IcoSetServer.cab file and the .gbc or .gbx file are properly referenced in the HTML source code for the main frame. This way the security and licensing information are available whenever a component is downloaded.

Delivering the Necessary Web Components to the Client

WebHMI is designed to operate with **Zero Install** and **Thin Client** philosophy. This means the client PC has nothing but Windows, Internet Explorer, or Netscape loaded. Thus, all necessary Web components must be remotely delivered and seamlessly installed. The type and number of components required for delivery on the client PC are determined by the content of the GraphWorX displays. For instance, a display with an embedded Trend Viewer requires delivery of the corresponding TrendWorX Viewer ActiveX Web component, but not the AlarmWorX Viewer ActiveX. On the other hand, a display with an embedded Alarm Viewer requires the corresponding AlarmWorX Viewer ActiceX but not the TrendWorX Viewer ActiveX.

The delivery process can be very slow in terms of network speed. To avoid wasting time, the first requirement is to deliver only the required component. The delivery process is strictly connected to the HTML code. Inside the HTML code, there are references to .dpl files or .cab files, the basic ways to deliver components. As a result, the building of the HTML page is a critical step. The Web Publishing Wizard looks inside the display to understand what components are needed on the client side and then builds the corresponding HTML code.

Netscape Navigator and Microsoft Internet Explorer neither process the HTML code in the same way nor deliver the component in the same standard format. Basically, Microsoft Internet Explorer requires components to be delivered using .cab file technology whereas Netscape Navigator requires .dpl files.

Client Station Requirement

In order to view a published GraphWorX display, a client PC must have a Web browser installed (e.g. Microsoft Internet Explorer or Netscape Navigator).

Developer Station Requirement

The developer station requires GraphWorX Version 6.1 or greater.

Web Server Station Requirement

The Web server PC must have the following installed:

- WebHMI
- ProcessView (with GenBroker)
- For Windows NT, Internet Information Server (IIS) or a Personal Web Server
- For Windows 2000 and Windows XP, Internet Information Server (IIS)

Multiple Display Support

Suppose that you have a main GraphWorX display (.gdf) file that is linked to other display files (e.g. each display contains pick actions, such as Load Display or Pop-up Window, that point to the other display files). When you publish the main display to an HTML file, you want all the links and references to the other dependent display files to be functional when the HTML file is downloaded to a client Web browser. The Web Publishing Wizard makes this possible by detecting all mutually linked display files, looping through all dynamic actions, and checking for pick actions in which a file name is specified as one of its parameters. The following pick actions are supported for multiple GraphWorX display Web publishing support:

- Load Display
- Embedded Window
- Popup Window
- Drag/Drop Load
- Set Aliases
- Alias Dialog (alias files specified through the **Set Aliases** dialog box in GraphWorX).

The LoadTabs display property is also supported for publishing multiple GraphWorX displays.

Embedded ActiveX Control Support

In dealing with a main GraphWorX display (.gdf) file that is linked to other display files (as described in the "Multiple Display Support" section above), suppose you included an ActiveX Control (e.g. Alarm Viewer ActiveX or Trend Viewer ActiveX) into each of the dependent display files. Also suppose that you generated and published an HTML file only from the main display, but you want this single HTML file to trigger the simultaneous download and installation of all ActiveX Controls embedded within all interlinked GraphWorX display files. The Web Publishing Wizard's embedded ActiveX Control support makes this possible, allowing you to view the ActiveX Controls in all displays from a single client Web browser.

Using the Web Publishing Wizard

The Web Publishing Wizard performs two basic operations:

1. The Wizard creates an HTML file based on a user-specified GraphWorX display (.gdf) file.
2. The Wizard then either "exports" (saves) the HTML file to a user-specified directory on the local drive or "publishes" (uploads) the HTML file to a user-specified Web server URL address (i.e. over the Internet or an intranet).

Starting the Web Publishing Wizard

To launch the Web Publishing Wizard in GraphWorX:

1. Load or create a GraphWorX display (.gdf) file.
2. After you have either created a new display file or opened an existing file, select **Save As** from the **File** menu. The **Save As** dialog box opens, allowing you to specify the name and location of the file you are about to save. Type a name for the new file in the **File Name** field. It is important that, when saving your file, you select "GraphWorX Displays without VBA (*.gdf)" in the **Save as Type** field, as shown in the figure below, because the Web Publishing Wizard does not support Microsoft Visual Basic for Applications (VBA). Click the **Save** button to save the current file.

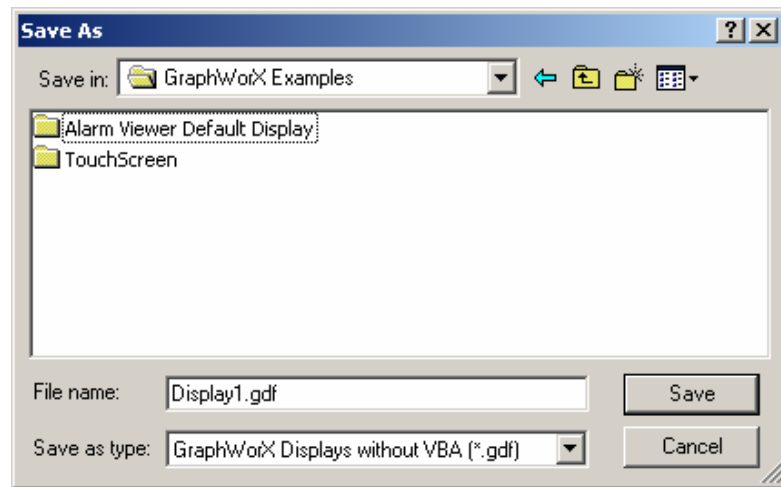


Figure 9.7. Saving the Display As a Non-VBA File

3. Select **Publish to HTML** from the **Tools** menu, as shown in the figure below.

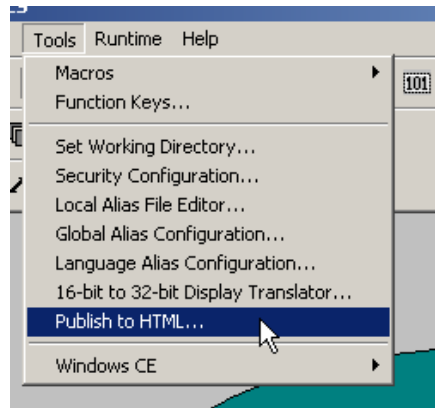


Figure 9.8. Starting the Web Publishing Wizard in GraphWorX

Note: If you try to run the Web Publishing Wizard with a GraphWorX display with VBA, you will get a warning message as shown in the figure below. If you click **Yes**, the Web Publishing Wizard automatically saves the display as a non-VBA file. If you click **No**, the publishing operation is cancelled.

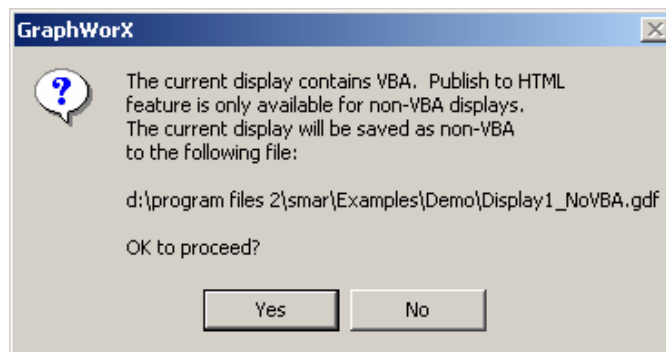


Figure 9.9. Warning Message About Displays With VBA

4. This launches the **Web Publishing Wizard** configuration dialog box, as shown in the figure below. This dialog box serves as the interface through which you export/publish GraphWorX display files to the Web.

Note: Both the **Publish to Web Server** and **Export Local Copy** actions can be performed at the same time.

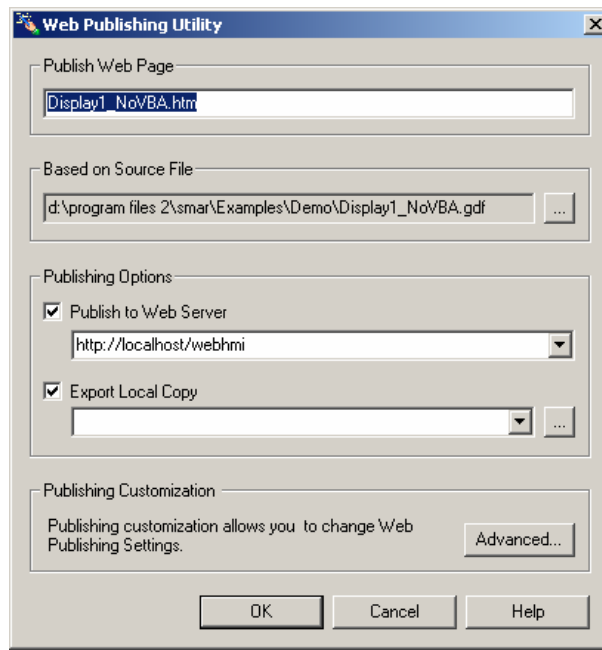


Figure 9.10. Web Publishing Wizard Dialog Box

Exporting a Display File Locally

To export a GraphWorX display to a directory on the local drive:

1. In the **Based on Source File** field of the Web Publishing Wizard, you must specify the name of the GraphWorX display (.gdf) file to be exported, as shown in the figure below. To choose a different display, click the ... button to the right to browse for a file. Select a file and then click **Open**. The directory path and the file name are shown in the text field.

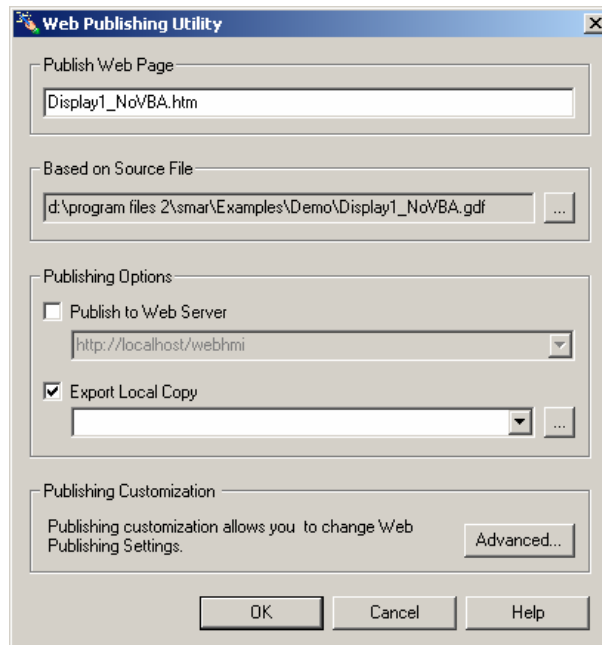


Figure 9.11. Specifying a GraphWorX Display File

2. In the **Publish Web Page** field of the Web Publishing Wizard, specify the name of the HTML file that will be created. The .gdf file name is filled in by default, but you can give the HTML file a different name.

- In the **Publishing Options** field, check the **Export Local Copy** check box and specify the local directory path name to which you want to export the HTML file. You can select a recently used path from the drop-down list, or click the ... button to browse for a destination directory, as shown in the figure below. Select the directory and click **OK**.

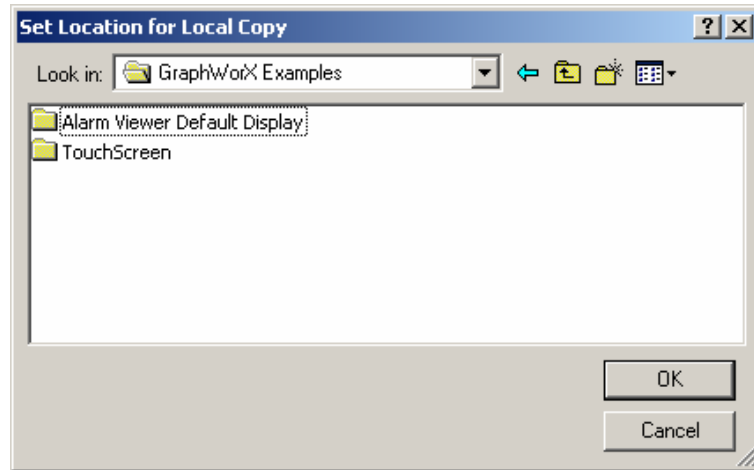


Figure 9.12. Setting the Local Directory

- The local directory pathway you selected appears in the **Export Local Copy** field of the Web Publishing Wizard, as shown in the figure below.

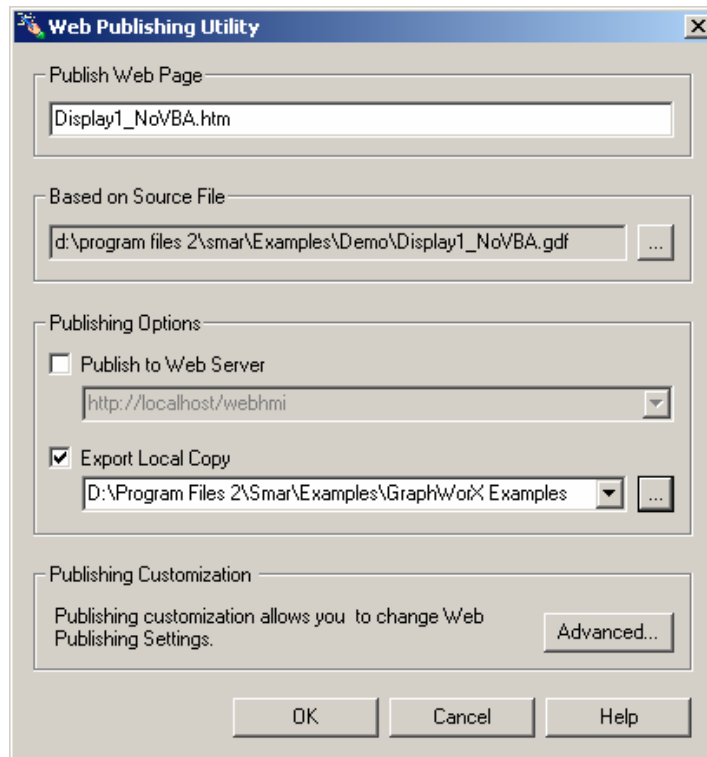


Figure 9.13. Export Local Copy Pathway Shown in the Wizard

- Click the **OK** button to generate the HTML file. The new HTML file is saved to the local directory pathway specified in the **Export Local Copy** field.

Note: Both the **Publish to Web Server** and **Export Local Copy** actions can be performed at the same time.

Publishing a Display File to a Web Server

To publish the HTML file to a directory on a Web server (i.e. over the Internet or an intranet):

1. In the **Based on Source File** field of the Web Publishing Wizard, you must specify the name of the GraphWorX display (.gdf) file to be exported, as shown in the figure below. To choose a different display, click the ... button to the right to browse for a file. Select a file and then click **Open**. The directory path and the file name are shown in the text field.

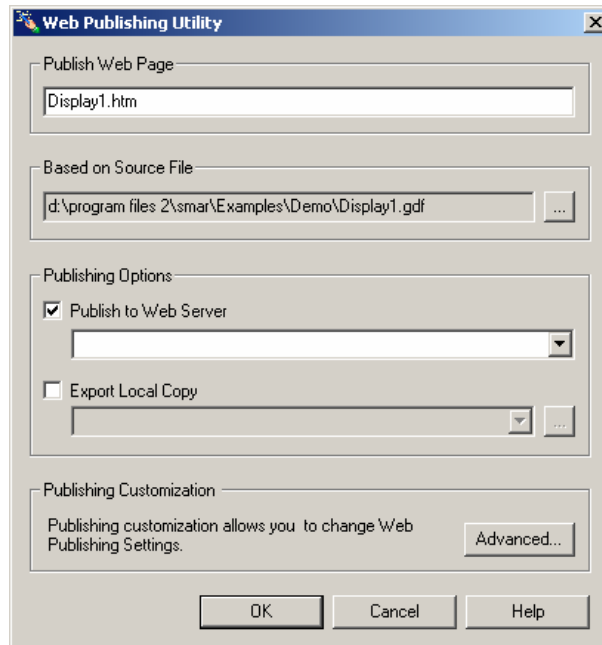


Figure 9.14. Specifying a GraphWorX Display File

2. In the **Publish Web Page** field of the Web Publishing Wizard, specify the name of the HTML file that will be created. The .gdf file name is filled in by default, but you can give the HTML file a different name.
3. In the **Publishing Options** field, check the **Publish to Web Server** check box and type in the URL address of your Web server with the complete directory indicating where you want to publish the HTML file, as shown in the figure below. In the sample Web server URL address shown below ("http://www.myserver.com/WebHMI/Samples"), the various components are:
 - **http://www.myserver.com:** IP address (server name) of the WebHMI Server
 - **WebHMI:** Name of the WebHMI server root directory
 - **Samples:** Name of the directory on the server to which HTML file will be saved
4. The Web server URL address now appears in the Web Publishing Wizard dialog box in the **Publish to Web Server** field of the Web Publishing Wizard, as shown in the figure below.

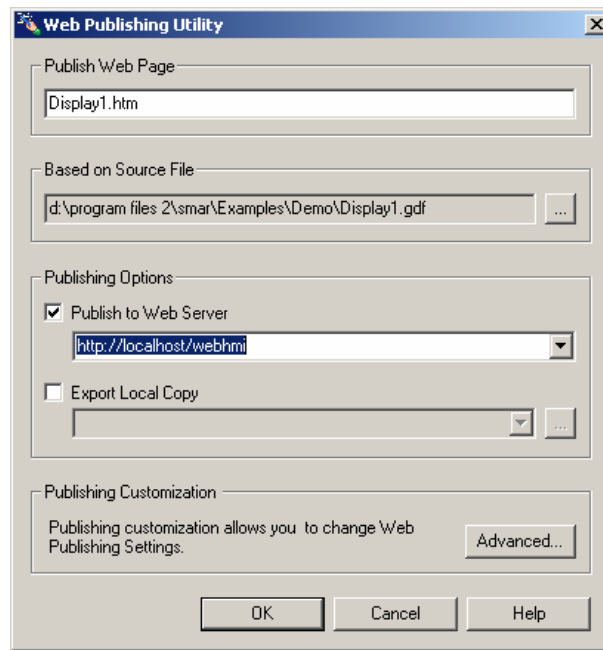


Figure 9.15. Publish to Web Server Location Specified in Web Publishing Wizard

5. Click the **OK** button to generate the HTML file. The new HTML file is uploaded to the Web server and then saved to the URL address path specified in the **Publish to Web Server** field.

Note: Both the **Publish to Web Server** and **Export Local Copy** actions can be performed at the same time.

Publishing Customization Options

The Web Publishing Wizard contains some customization options for exporting/publishing HTML files. Clicking the **Advanced** button on the Web Publishing Wizard dialog opens the **Web Publishing Properties** dialog box, as shown in the figure below, which contains the following tabs:

- General Settings
- Server Switching Support
- Screen Resolution Settings
- Publishing Options

General Settings

The **General** tab of the **Web Publishing Properties** dialog box, shown in the figure below, allows you to specify the default URL address for the WebHMI Server. This WebHMI root URL path is used as the default path when only **Export Local Copy** is selected on the Web Publishing Wizard dialog box. (If **Publish to Web Server** is selected on the Web Publishing Wizard dialog box, you can specify a different URL path for the WebHMI Server.)

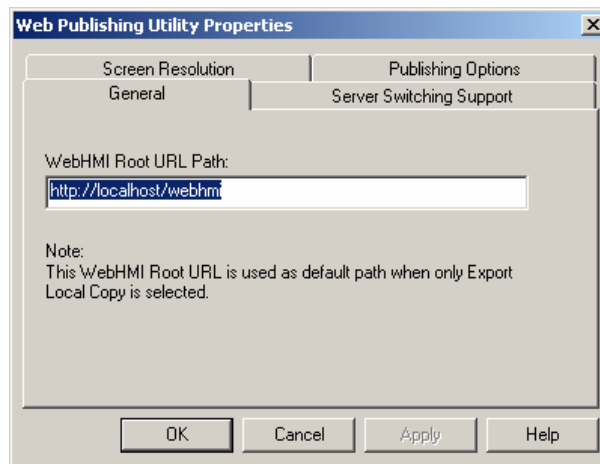


Figure 9.16. General Settings

Server Switching Support

The **Server Switching Support** tab of the **Web Publishing Properties** dialog box, shown in the figure below, allows you to enable or disable GenBroker support for the exported/published HTML file. Here you can specify which GenBroker configuration file (.gbc or .gbx) will be activated. The GenBroker configuration file establishes the settings for OPC data communications between the clients and the Web server.

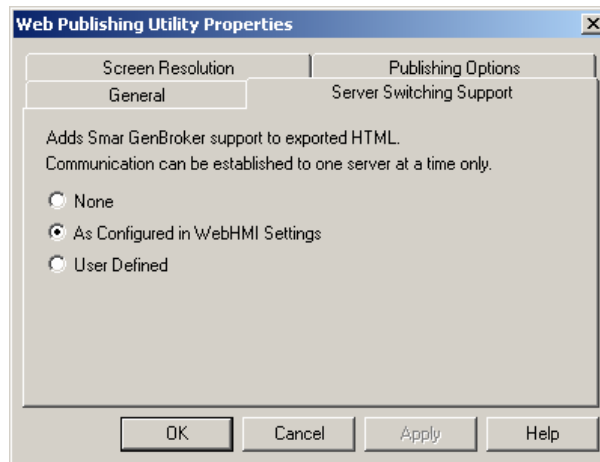


Figure 9.17. Server Switching Support Settings

There are three available options for GenBroker support:

- **None:** GenBroker support is not active.
- **As Configured in WebHMI Settings:** This setting uses the default URL address of the GenBroker configuration (.gbc or .gbx) file as specified in the WebHMI installation.
- **User Defined:** When this option is selected, the **GenBroker Configuration File URL** field becomes available, allowing you to specify the URL address of the GenBroker configuration (.gbc or .gbx) file.

Clients can receive data from different WebHMI servers, but communication can be established only to one server at a time. The server to which a client connects depends on the type of information that the client requests. If Server A, for example, does not contain the components necessary to answer the client's request, the request will be forwarded to Server B, or Server C, and so on.

Note: For information about configuring GenBroker settings, please see the GenBroker Configurator Help documentation.

Screen Resolution Settings

The **Screen Resolution** tab of the **Web Publishing Properties** dialog box, shown in the figure below, determines the screen resolution and size of the GraphWorX Viewer ActiveX control, which is referenced in the generated HTML file and then downloaded to a client PC when the HTML file is viewed in the client's Web browser. You can specify the screen size in the **Width** and **Height** fields in terms of pixels (px) or percentage (%). Click the **Resolutions** button to select from a pop-up menu of standard screen resolutions, as shown in the figure below.

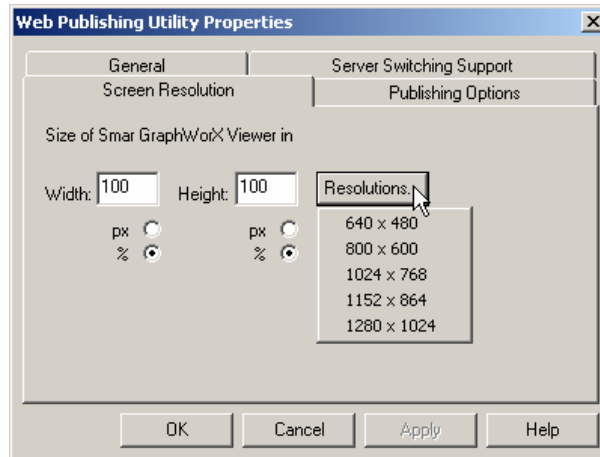


Figure 9.18. Screen Resolution Settings

Publishing Options

The **Publishing Options** tab of the **Web Publishing Properties** dialog box, shown in the figure below, enhances the publishing process. By default, the Web Publishing Wizard provides support for related files detection (for more details, see the **Multiple Display Support** section). The multiple display detection could be a rather lengthy process. You can speed up the process by disabling the **Enable Multiple Display Support** option. This is especially useful, if you have already published your project files to the Web server, and now you want to update display you have changed.

The multiple display detection mechanism ensures that the Web page will be published properly for the source file. Disabling the **Multiple Display Support** may cause publishing of an incomplete web page, which may result in unexpected behavior. Therefore, it is suggested to disallow publishing of the Web page. You can do so by enabling the **Publish Display File Only** option.

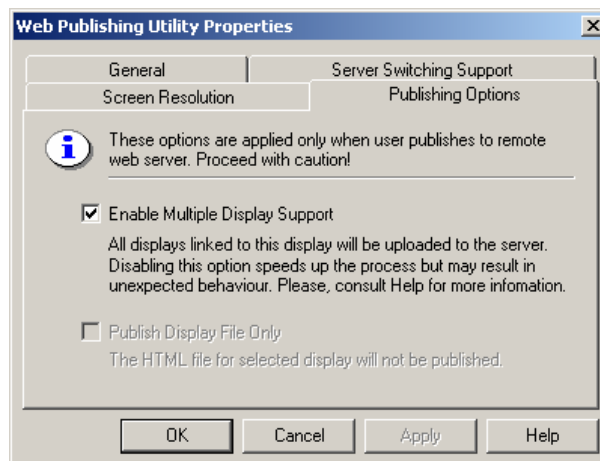


Figure 9.19. Publishing Options

Viewing Locally Exported HTML Files

To view an HTML file that was exported to a local directory, simply browse to the local directory pathway and click on the file to open it in your Web browser. You can also open your Web browser and copy the directory pathway of the file into the **Address** field of the Web browser, as shown in the figure below. A complete directory path is, for example, "C:\Documents and Settings\Administrator\My Documents\Web Publishing Example\ChemFood_BeanRoaster.htm." Be sure to include the name of the HTML file that you have exported and the file extension ".htm."



Figure 9.20. Viewing the Exported HTML File in a Web Browser

Viewing Published HTML Files

Before a published HTML file can be downloaded from a WebHMI server, you must set GenBroker active on the WebHMI server node using ProcessView Tray:

1. Launch ProcessView Tray from the Windows **Start** menu by selecting **Programs > System32 > ProcessView > Tools > ProcessView Tray**.
2. When ProcessView Tray opens, the triangle icon will appear in the Windows tool tray. Click on the triangle and select **GenBroker > Start** from the pop-up menu, as shown in the figure below. This activates the GenBroker Server.

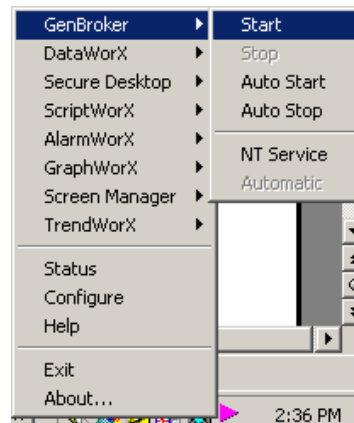


Figure 9.21. Starting the GenBroker Server on the Server Node

3. Once the GenBroker Server is running, you can view the HTML file that you have published to the Web server from any client PC's Web browser. To view the published HTML file, open your Web browser and copy the URL address of the file into the **Address** field of the Web browser, as shown in the figure below. A complete address is, for example, "http://www.myserver.com/webhmi/Samples/ChemFood_BeanRoaster.htm." Be sure to include the name of the HTML file that you have published and the ".htm" file extension. The GraphWorX display file is referenced in the HTML code so the display can be viewed as a Web page. The client's Web browser simply downloads the HTML file in which the .gdf file is referenced. All the client needs is a Web browser; it is not necessary to have ProcessView installed on client. The Web page is downloaded from the Web server across the Internet/intranet and appears in the client's Web browser window. The display is real-time, just as if you were viewing the runtime display in GraphWorX on the server machine; the OPC tag values change dynamically in the display.

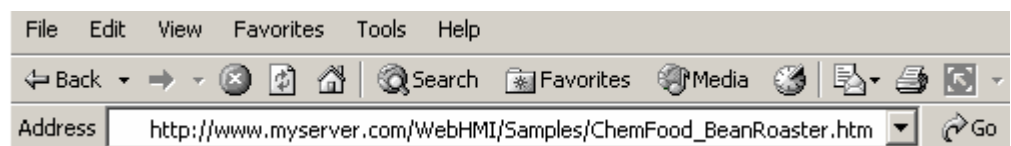


Figure 9.22. Viewing the Published HTML File in a Client Web Browser

Configuring for Windows CE in ProcessView

All display files for Pocket ProcessView are first configured on a desktop PC using a ProcessView application - such as GraphWorX, TrendWorX, and AlarmWorX - and then saved in a format that is compatible with Pocket PC and Windows CE. A display file is then downloaded to the Pocket PC or Windows CE device to be viewed during runtime. Once you open the display file in the corresponding Pocket ProcessView application (i.e. Pocket GraphWorX, Pocket TrendWorX, or Pocket AlarmWorX) in runtime mode, you can execute all runtime functional dynamics that have been added during configuration. With a few exceptions, Pocket ProcessView applications have the same basic runtime functionality as their counterpart ProcessView applications.

Configuring for Windows CE in ProcessView

To configure for Windows CE in GraphWorX, TrendWorX, and AlarmWorX, select **Windows CE - Configure for Windows CE** from the **Tools** menu, as shown in the figure below. If you have just installed Pocket ProcessView, this will be checked by default.

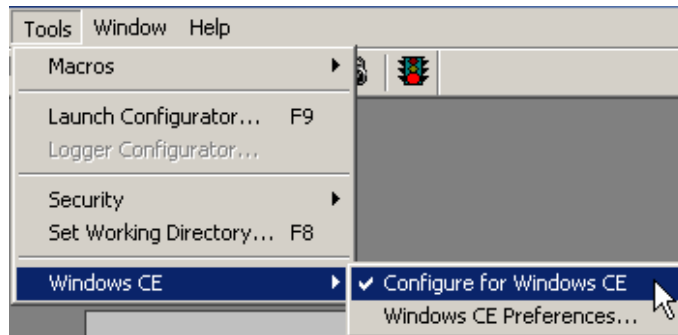


Figure 9.23. Configuring for Windows CE

Once you have configured your display, select **Save As** from the **File** menu. This opens the **Save As** dialog box, which allows you to save your configuration as a display for Windows CE file. Give the file a name and select one of the following file types from the **Save As Type** box:

- In GraphWorX, select either "GraphWorX Displays for WinCE (*.gdc)" or "GraphWorX Templates for WinCE (*.gdc)."
- In AlarmWorX, select "AWXview CE File (*.awv)."
- In TrendWorX, select "TWXviewerCE File (*.tce)."

Click the **Save** button. Upon saving your configuration file for Windows CE, features that are not supported by Pocket PC or Windows CE will be removed from the Pocket ProcessView display.

Note: When opening a Pocket ProcessView display in ProcessView, you must specify the proper Windows CE file extension.

Remote Connections

In order to connect remotely to OPC servers over a network, you must install and use GenBroker. Please see the GenBroker Help documentation for more information about configuring GenBroker.

Downloading ProcessView Configuration Files to Your Pocket PC

Once you have configured your application in ProcessView and saved the Pocket ProcessView display file, you can use the file Download to Windows CE Utility in ProcessView to download configuration files from GraphWorX, TrendWorX, and AlarmWorX to your Pocket or CE device. When developing a configuration file for a Pocket ProcessView application, this feature allows you to download the configuration file to a Windows CE or Pocket PC device. The file download function uses Microsoft ActiveSync to connect to the CE device.

Setting up the Download

The download to CE tool requires modules on both a desktop or workstation PC as well as a Pocket PC or CE device, because Microsoft ActiveSync services are used for connecting and authenticating the CE device.

Configuring the Desktop

The desktop is the only part that has to be configured. If the file download tool is installed properly, it will be listed in the ActiveSync Manager list of ActiveSync modules in the **Sync Options** tab of the Windows **Options** dialog box. You must enable the file download ActiveSync module in the ActiveSync Manager by checking the **Download** box, as shown in the figure below.

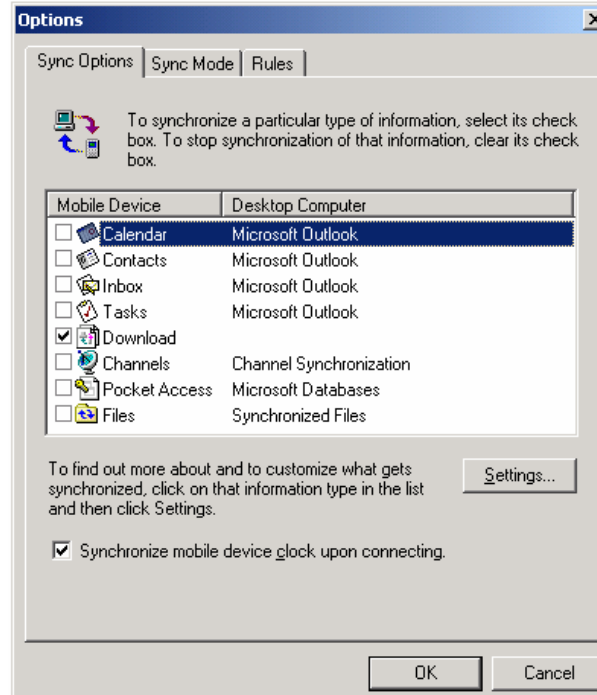


Figure 9.24. Windows Options Dialog Box

An association with the CE device must be created to synchronize the download ActiveSync modules, as shown in the figure below. When the CE device responds without any problems, the download synchronization module is active.

When enabled, the file download tool immediately downloads the configuration file for Windows CE to the CE device. When disabled, the configured file will only be saved.

For more information about Microsoft ActiveSync services, please refer to the Microsoft ActiveSync help documentation.

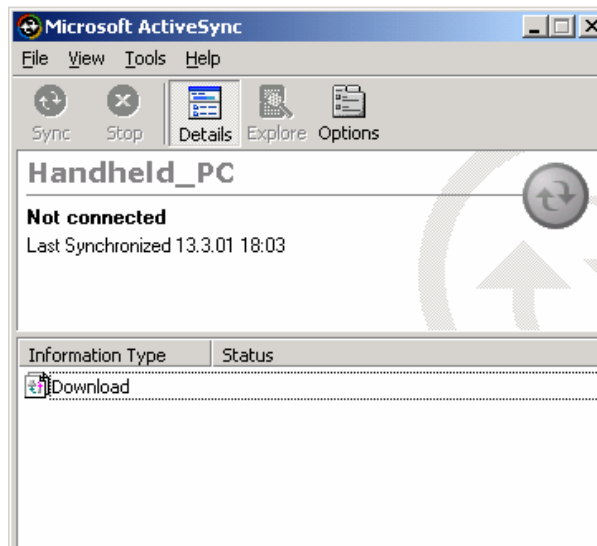


Figure 9.25. Microsoft ActiveSync

File Download Application Configuration

GraphWorX, TrendWorX, and AlarmWorX have their own version of the file download tool. In these applications, choose **Tools > Windows CE > Windows CE Preferences**, as shown in the figure below.

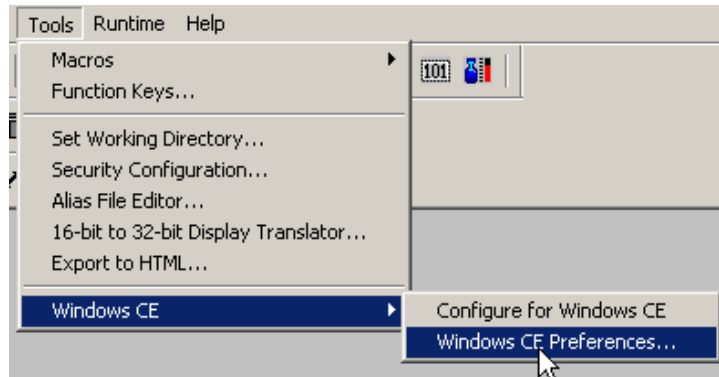


Figure 9.26. Starting the Windows CE Download Tool

This opens the **Windows CE Preferences** dialog box, which allows you to enable or disable the file download tool. To enable this tool, check the **Download On Save** check box. Then select the destination directory on the CE device by entering the path name in the **Download Directory** field, as shown in the figure below.

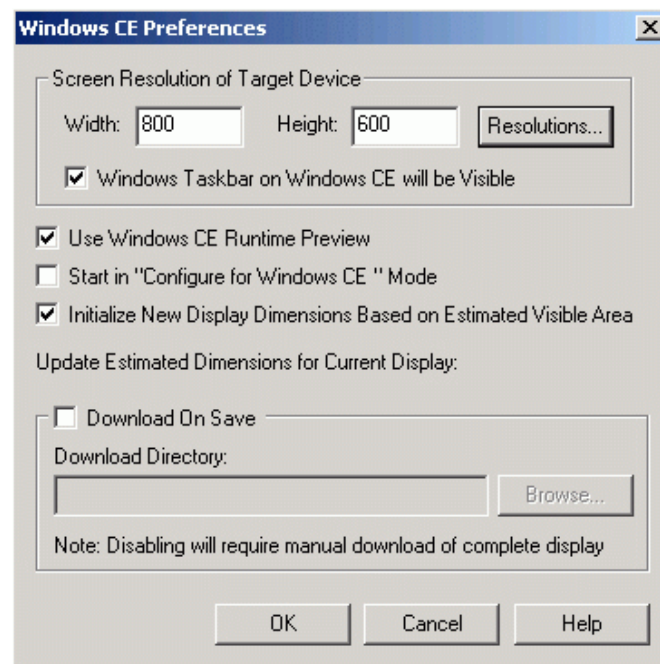


Figure 9.27. Setting Windows CE Preferences

You can also click the **Browse** button to select the destination directory. Clicking the **Browse** button opens the **Browse for Folder** dialog box, which shows the CE device directories, as shown in the figure below. A warning message box will be displayed when the file download ActiveSync module is inactive.

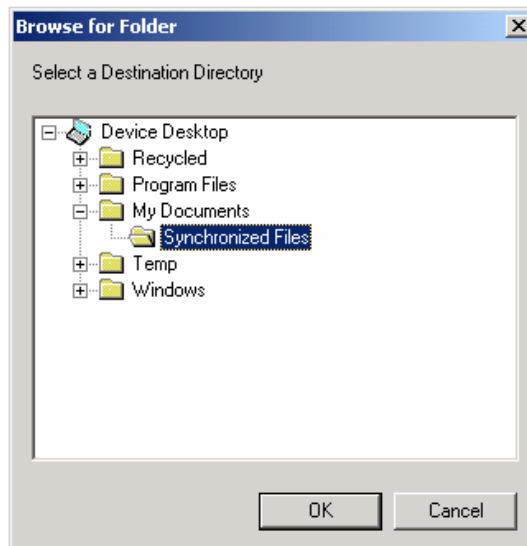


Figure 9.28. Browsing for a Folder on the CE Device

To establish a connection with the CE device, click **OK**, as show in the figure below.

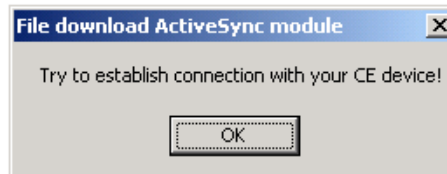


Figure 9.29. File Download ActiveSync Module

You can also use the **Windows CE Preferences** dialog box to configure other settings for the CE device. You can set the resolution of the target device by clicking the **Resolution** button. This displays a list of resolutions from 200 x 320 up to 1024 x 768.

Downloading the File to the CE Device

When you have finished configuring a display file and would like to save it to Windows CE, choose **Save As** from the **File** menu. This opens the **Save As** dialog box.

- In GraphWorX, enter the file name and select either "GraphWorX Displays for WinCE (*.gdc)" or "GraphWorX Templates for WinCE (*.gdc)" from the **Save As Type** box.
- In AlarmWorX, enter the file name and select "AWXview CE File (*.awv)" from the **Save As Type** box.
- In TrendWorX, enter the file name and select "TWXviewerCE File (*.tce)" from the **Save As Type** box.

The Smar **Import File** dialog box will be displayed, indicating that the files are downloading to the Windows CE device.

Starting Pocket ProcessView Applications on Your Pocket PC

Once you have downloaded your GraphWorX, AlarmWorX, or TrendWorX configuration files from your desktop PC to your Pocket PC or CE device, you can run the applications on your Pocket PC. The Pocket ProcessView client applications (Pocket GraphWorX, Pocket AlarmWorX, and Pocket TrendWorX) are located in the **SMAR/Pocket ProcessView** directory on your Pocket PC. Open the application you wish to run, and then select **Open** from the **File** menu. Browse for the desired .gdc, .awv, or .tce file. An example using Pocket GraphWorX is provided below.

1. Open Pocket GraphWorX from the **SMAR/Pocket ProcessView** directory on your Pocket PC. In Pocket GraphWorX, select **Open** from the **File** menu, as shown in the figure below.



Figure 9.30. Opening a Pocket GraphWorX Configuration File

2. Browse for a Pocket GraphWorX display (.gdc) file, as shown in the figure below. In this example, the **chiller1.gdc** file is selected.

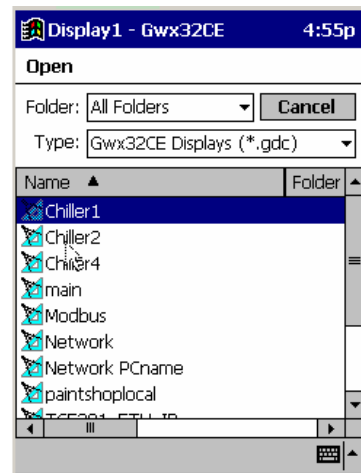


Figure 9.31. Browsing for a Display File

3. The display file opens in Pocket GraphWorX, as shown in the figure below. The runtime operations of Pocket ProcessView enable you to see real-time OPC data values in the display.



Figure 9.32. Viewing the Display in Pocket GraphWorX

Note: The procedure for opening and viewing files in Pocket AlarmWorX and Pocket TrendWorX is similar to the procedure described above for Pocket GraphWorX.

Scripting Engine and Script Editor for VBScript and JScript

GraphWorX has a Visual Basic for Applications (VBA) integrated editor, which allows writing of VBA Script. This is a powerful tool that allows easy expansion of GraphWorX functionality. In order to run a VBA Script, a PC must have the VBA Runtime Engine. This engine cannot be delivered through the Internet on a thin-client machine because it takes a lot of disk space and requires a long download time. Because of this, in the past Smar has decided to remove VBA support from WebHMI and the other "lightweight" products, such as Pocket GraphWorX. So it is impossible to use the power of VBA and the advantage of the thin-client technology of WebHMI. Many users develop two different sets of displays: one set with VBA for the desktop workstation, and another set without VBA for the WebHMI thin client. This requires additional work needed to strip out the VBA code and redesign the display.

As a solution to the problem described above, GraphWorX features a new scripting engine that can parse VBScript and Jscript code. This engine is lightweight and can easily be distributed over the Internet to any thin-client machine. When you plan to use the scripting technology for enhancing GraphWorX features and also plan to use thin-client technology, such as WebHMI, you should use the new embedded scripting engine and write your code either in VBScript or Jscript instead of Visual Basic for Applications. You can now design one unique display with script and WebHMI support, without having to manage multiple displays.

The new scripting engine for VBScript and Jscript does not have all the power of a full VBA engine, so some features are not supported. There are many limitations, but the largest one is probably the fact that it is impossible to handle events coming from ActiveX or other controls embedded inside a GraphWorX display.

The new scripting engine is able to handle the most important display events, and all the pick actions, but it is currently impossible to fire new events.

An additional limitation is that VBScript and JScript do not support forms, so it is impossible to create user interface elements different from a simple message box or input box.

Displaying the Script Editor

The integrated Script Editor appears as a toolbar in the GraphWorX display. It can be dragged around the display as a floating window or it can be docked in place on one of the four display sides. You can resize the toolbar (both docked and floating) at any time by dragging its border. The Script Editor is an integrated toolbar, so you do not need to "open" it. Instead you just "show" or "hide" it by selecting **Toggle Script Toolbar** from the **View** menu, as shown in the figure below.

Note: You can also use the convenient shortcut key combination **CTRL+ALT+J**, or select **Toolbars** from the **View** menu and check the **Script** check box on the **Show Toolbars** dialog box. In addition, you can toggle the Script Editor toolbar by clicking the "pencil" button on the **Main** toolbar.

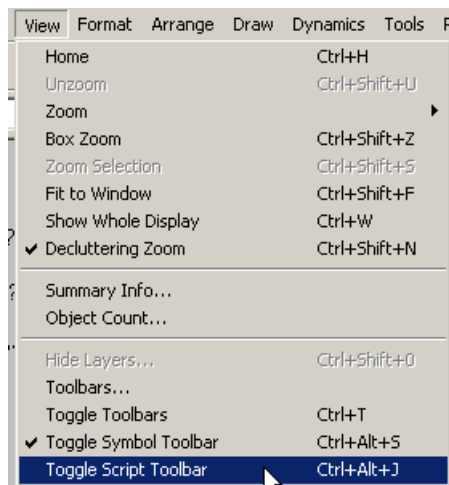


Figure 9.33. Starting the VBScript / Jscript Events Editor

The **Script Editor** is displayed, as shown in the figure below. The editor is divided into a left panel and a right panel. The left panel is the script browser, and the right panel shows the code for the selected script. Each script shown in the left panel browser is identified as VBScript (**VB**) or JScript (**JS**), as shown in the figure below. You can change the size of the left and right panel of the script by dragging the splitter bar between them.

The Script Editor has two main editing “modes”: **event mode** and **object mode**. When the Script Editor is in event mode, the left panel lists all the display events that can be scripted. When the Script Editor is in object mode, left panel lists the scripted objects in the display. You can switch the script editor mode by clicking on the left panel header. The figures below show the event and object modes.

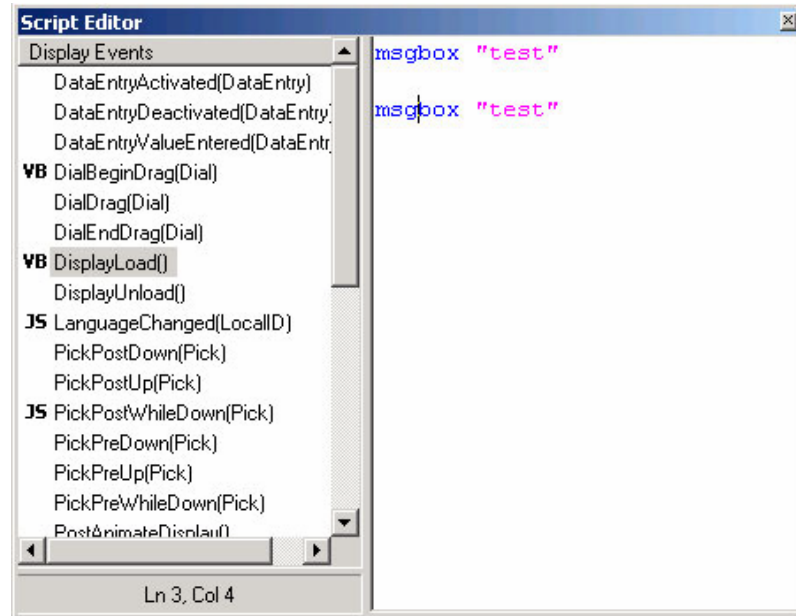


Figure 9.34. Script Editor in Display Events Mode

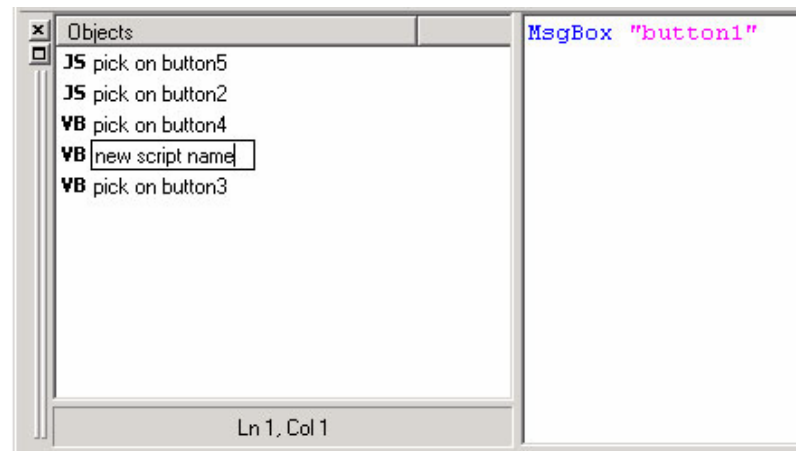


Figure 9.35. Script Editor in Objects Mode

Each script and its corresponding object is synchronized in the display. Therefore, if you select an object in the display that has a script associated with it, then the script is automatically highlighted in the Script Editor. Conversely, you select a script in the Script Editor, then the corresponding object is also selected in the display.

Event Mode

When the script editor is in **event mode**, then the left panel lists all the display events that can be scripted. For each event, it is possible to write a script to handle it.

A complete list of these events can be found in the GraphWorX documentation.

To associate a script with an event, right-click the event in the script browser (left panel). A pop-up menu appears, as shown in the figure below. You can then choose the language that you want to use to handle the script (VBScript or JScript).

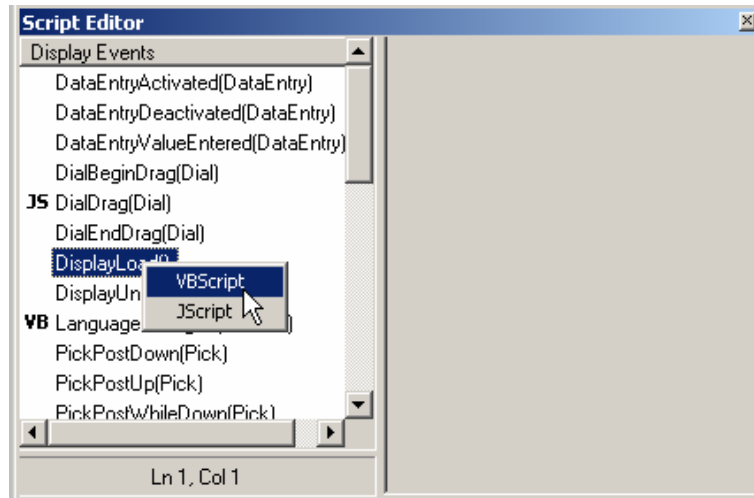


Figure 9.36. Selecting the Script Type

Once you have chosen the script type then you can start editing the script.

The next time you open the pop-up menu, the script type is indicated by a check mark. You can change the script type at any time.

Note: If you uncheck the item, you will delete the script. A dialog will ask to confirm before deleting the script, as shown in the figure below.

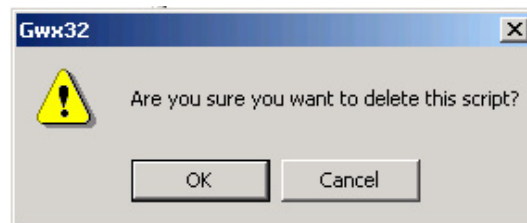


Figure 9.37. Deleting a Script

Object Mode

When the Script Editor is in object mode, you can edit the script associated with an object. It is possible to associate a script with an object through the “pick action” dynamic. A pick action is a special kind of dynamic that is executed in each time you click on the object. When the script editor is in object mode, the left panel lists the scripted objects in the display. A **scripted object** is any GraphWorX object that has a “run script” pick action associated with it and has the type “VBScript” or “Jscript.” The Script Editor never shows the objects with the regular VBA script pick action. The VBA script can be edited only with the standard VBA Editor.

To create a new script and associate it to an object you have to add a “pick action” dynamic to the object:

1. In the **Pick** tab of the Property Inspector, choose the **Run Script** pick action and type a name for the script in the **Script Name** field. If you omit the script name, GraphWorX will automatically use the object name property of the pick object. (If it also is omitted, then GraphWorX will automatically associate a name with this script.)

- Select the type from the **Script Type** drop-down list, as shown in the figure below. Only VBScript and JScript can be edited with the Script Editor.

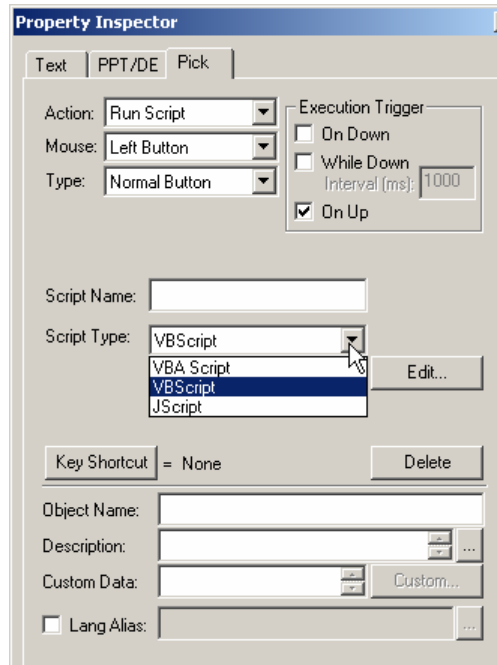


Figure 9.38. Choosing a Script Type for the Run Script Pick Action

- Click **OK**. When you choose VBScript or JScript, a new script is generated in the **Script Editor**. The script name automatically appears in the Script Editor. If the Script Editor was in event mode, then it automatically switches to object mode and selects the script that you have just created. Can then start to write the script.

Changing the Script Name

You can change the name of a script at any time either through the object property page or directly in the Script Editor. When the editor is in object mode, you can rename the currently selected script by clicking on the script name inside the script browser (left panel).

Note: It is not possible to change the script name when the editor is in event mode. In other words, you cannot change the default name for the events.

Basic Text Operation

When you right-click on the script (right) panel, a pop-up menu appears, as shown in the figure below. This menu contains all the basic text operations: **Undo**, **Cut**, **Copy**, **Paste**, **Delete**, and **Select All**.

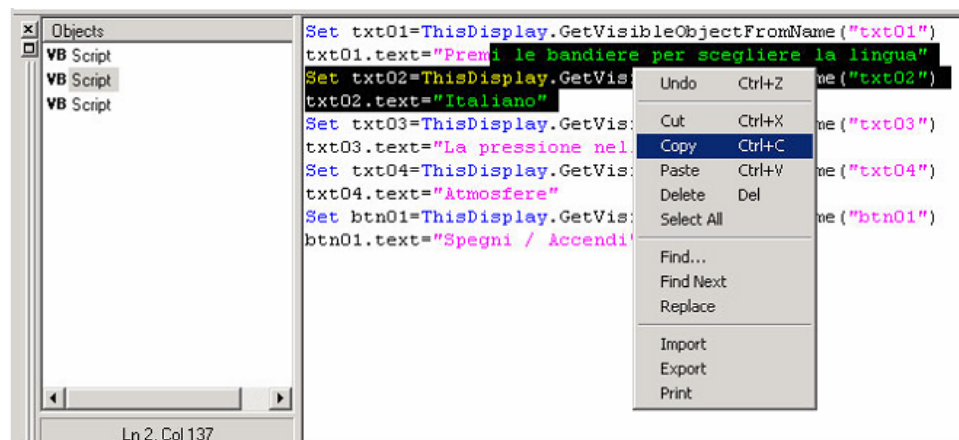


Figure 9.39. Editing Text in the Script Editor

Text Search and Replace

Right-clicking the script (right) panel and selecting **Find**, **Find Next**, or **Replace** allows you to do text search and replace functions, as shown in the figure below. You can choose few advanced options, which allow you to match the whole word only and to match the case. You can also specify whether to search down or up and to replace only the first instance of the searched text or all of the instances present in the script text. You can keep searching in the text by clicking **Find Next** on the pop-up menu.

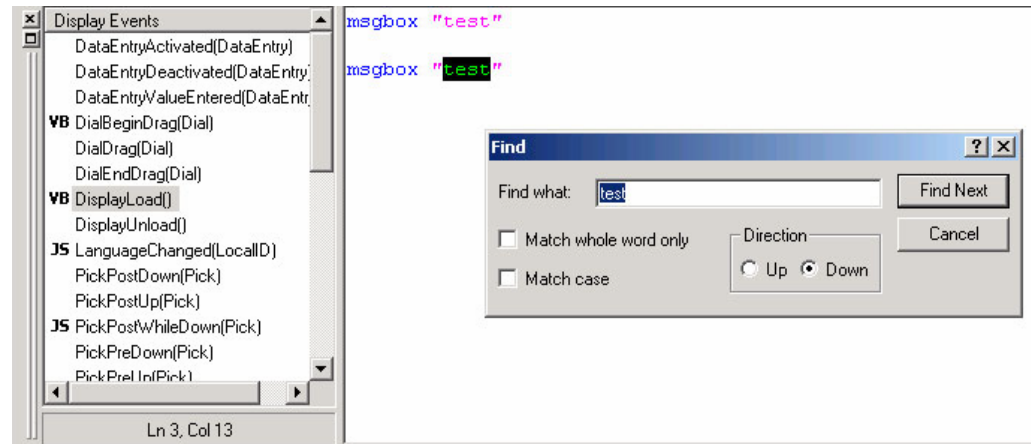


Figure 9.40. Basic Search and Replace

Importing and Exporting Scripts

When you right-click the script (right) panel and select **Import** or **Export**, you can import and export the script text from or to a separate file. You can import text into the script editor from any text file. You can also import text from the standard VBScript and JScript files (*.vbs and *.js), as shown in the figure below.

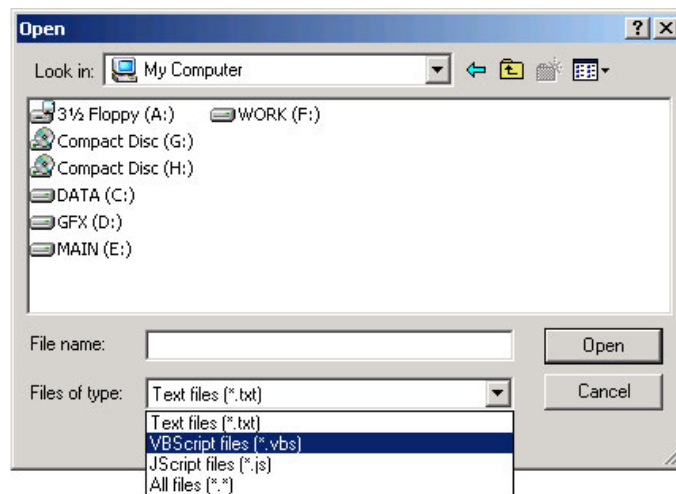
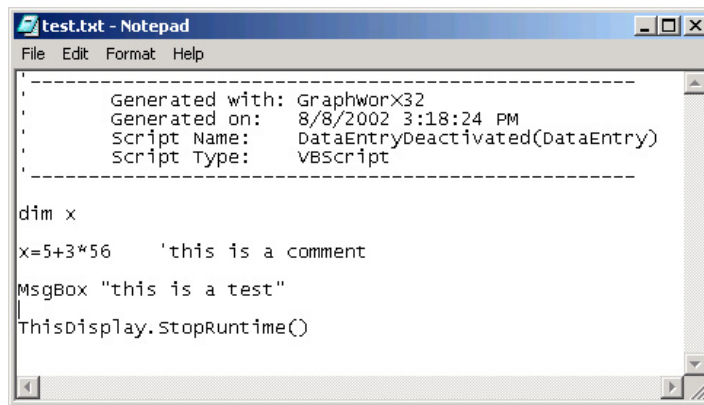


Figure 9.41. Importing and Exporting Scripts

You can export your current script to a text file, a VBScript file or a JScript file. Each time you export a script, GraphWorX automatically adds a header that describes the file content and the time it was generated, as shown in the figure below. Notice the application header with the timestamp, the script type, and the script name. In this case, the script was associated with an event, so the header has the event name as a script name.



```

Generated with: GraphworX32
Generated on: 8/8/2002 3:18:24 PM
Script Name: DataEntryDeactivated(DataEntry)
Script Type: VBScript

-----

dim x
x=5+3*56 'this is a comment
MsgBox "this is a test"
ThisDisplay.StopRuntime()

```

Figure 9.42. Script Exported to Text File

Print Scripts

You can print script directly from the script editor by right-clicking the script (right) panel and selecting **Print** from the pop-up menu. This opens the standard Windows Print dialog box where you can configure the printer settings. You can choose the printer that you wish to use, and you can also choose the number of copies that you want to print. There is no support for printing only a range of page or only the selected text. The support for “collate” is also disabled.

Preferences and Settings

The Script Editor is designed to be customizable for users with special needs. The customization settings can be configured in the **Script Editor** tab of the **Application Preferences** dialog box by selecting **Application Preferences** from the **Format** menu. You can change the following Script Editor settings, as shown in the figure below:

- Font face, size and style
- VBScript keywords and constants
- JScript keywords and constants
- Color for keyword, constants, string, number, and comment
- Enable or disable the syntax coloring
- Enable or disable the automatic case change feature

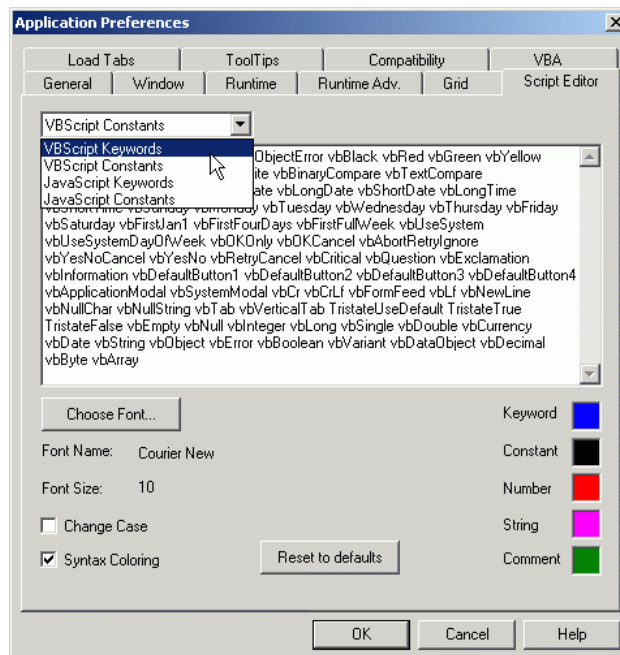


Figure 9.43. Application Preferences: Script Editor Tab

Runtime Environment

This section discusses the GraphWorX runtime environment. The runtime environment lets you view multiple windows of displays on screen. You can view any combination of the ProcessView windows applications (i.e. TrendWorX and AlarmWorX as well as other applications such as Microsoft Excel.)

The runtime environment functions according to how you configure the runtime parameters in GraphWorx configuration mode.

Starting Runtime

Menu Bar Option

To start runtime mode in GraphWorX, select **Runtime** on the menu bar.

Shortcut on Desktop

1. Right-click on the shortcut for GraphWorX to open the **Properties** menu.
2. Select **Shortcut > Target** and type in "runtime" to enter runtime mode from the shortcut.

Command Line Option

1. Select **Run** on the **Start** menu. Type in the path for GWX32.exe.
2. Specify a display and type in the path name for a display in GraphWorX ("display.gdf") to launch the runtime mode with a display.

When you enter runtime mode, the menu bar changes to display the following runtime menus:

- File
- View
- Tools
- Configure: Exits runtime mode and returns to configuration mode.
- Help

Runtime File Menu

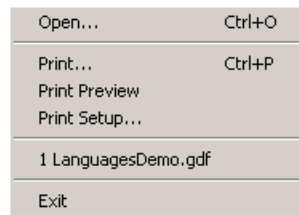


Figure 10.1. Runtime File Menu

File Functions

File - Open

To load displays during runtime mode:

1. Select the desired window if you have multiple windows running.
2. Select **Open** from the **File** menu. This opens the **Open File** dialog box.
3. Select a display file and click **OK**. The display appears in the selected window.

Printing the Screen

While in the runtime environment, you can initiate a print screen action to print the entire screen. This function prints the screen to the currently configured Windows printer. To print the full screen:

1. Select the **Print** from the **File** menu. This opens the **Print** dialog box.
2. Complete the required fields and click **OK**.
3. The screen is printed immediately.

Print Preview

To preview the display prior to printing:

4. Select Print Preview from the File menu.
5. The menu bar provides options to preview the current page, previous page, next page, or two pages at once. It also lets you zoom in and out of the display.
6. Click Close to exit the preview mode to the full screen mode.

File - Print Setup

To configure printer properties:

7. Select Print Setup from the File menu to select the printer, paper size, source, and orientation. This opens the Print Setup dialog box.
8. Define the required parameters and click OK.

Runtime View Menu

The **View** menu in runtime mode provides the same options as the **View** menu that is available in configuration mode.

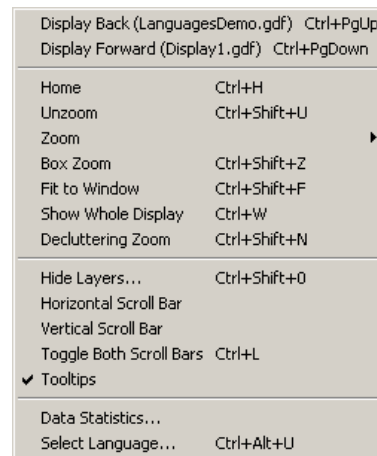


Figure 10.2. Runtime View Menu

Data Statistics

Selecting **Data Statistics** from the **View** menu shows runtime information in the Statistics Viewer dialog. This dialog is implemented in the GenClient/Oleexpress DLL library and it is available throughout the ProcessView applications.

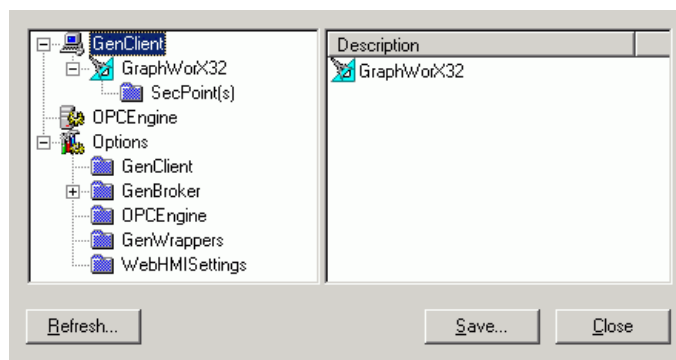


Figure 10.3. Statistics Runtime Viewer

Display History

GraphWorX maintains a history of the last 50 displays. You can browse through the display history using the commands **DisplayBack** and **DisplayForward** on the **View** menu. These commands are available as pick commands and as OLE Automation methods. The display history feature is similar to Microsoft Internet Explorer's back/forward feature.

Note: The display file history (forward/back commands) remembers the initial alias settings specified when a display is opened (see description of the runtime aliasing feature), and resets those aliases when you go back to that display in the file history.

Runtime Security

Access to the runtime environment of GraphWorX is restricted by the Security Server. Security levels are set initially in the project file and determine how much accessibility operators have during runtime. To login to runtime:

1. Select **Security Login** from the **Tools** menu. The **Security Login** dialog box opens.
2. Enter the user name and password in the appropriate fields, and then click **OK**.

Where there was insufficient security to view the current display, previous versions of GraphWorX shut down that display, and if there were no other windows or menus in the current display, the user was locked out by the Security Server without a way to log in as another user.

In the latest version of GraphWorX, if a display is shut down by the Security Server, a separate display file called "Login.gdf" automatically opens, as shown in the figure below, invoking the Security Login dialog box. In addition, GraphWorX displays the Security Login dialog in all cases where there is insufficient security.

Note: Please refer to the **Security Configurator** documentation for more detailed information on security.

Runtime Data Entry

During runtime mode, you can quickly activate or enter data-entry fields by using the **TAB** key.

TraceWorX Support

Using a technology that has been incorporated into all Smar products, TraceWorX provides online diagnostics and tuning of applications running in the ProcessView system. TraceWorX is designed expressly for systems integrators, OEMs and customers who want to have tools for doing their own troubleshooting and diagnostics.

TraceWorX tracks the runtime activity for each ProcessView application and logs the runtime data to a log file based on user-configured trace levels. The log file provides a thorough, color-coded report detailing all activity for the application, including the time, the date, the severity level, and a description of the event or problem.

TraceWorX also features several options for reporting issues to technical support. If you are experiencing problems with any applications, the log file deployment options, such as compressing and e-mailing log files, are ideal for tracking and archiving data and sending detailed reports to technical support. Developers can use these reports to identify the source of the problems.

TraceWorX can be activated by the following registry keys for each GraphWorX component. The **ReleaseTraceLevel** can be set between 0 (full trace information) and 1000 (turned off). When it is active, a log file is generated either in the component's current directory, or in the subfolder of "Documents and Settings." Search for the *.log.xml document.

```
[HKEY_LOCAL_MACHINE\SOFTWARE\ICONICS\GWX32]
"ReleaseTraceLevel"=dword:00000000
[HKEY_LOCAL_MACHINE\SOFTWARE\ICONICS\GWXview32]
"ReleaseTraceLevel"=dword:00000000
```

For more information about TraceWorX, please see the TraceWorX Help documentation.

OLE Express Compatibility

GraphWorX can imitate OLEExpress 6.0 compatibility behavior. The patch fixes the jumping slider problem, but it also introduces the potential problem of data disintegrity. The feature is turned off by default.

Note that it can be optionally disabled for GWX32.exe by the following registry setting:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\ICONICS\GWX32]
"CompatibilityOleexpress60Write"=dword:00000000
```

and for GWXview32.ocx by the following registry setting:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\ICONICS\GWXview32]
"CompatibilityOleexpress60Write"=dword:00000000
```

The feature immediately updates other dynamics in the same display, so you can place a process point next to the slider and view the precise value you enter.

Note:

VBA Wizards

Introduction to VBA Wizards

Visual Basic for Applications (VBA) Wizards are GraphWorX objects with Visual Basic code behind them. The code is run either in design mode to help to configure GraphWorX object or in runtime mode to execute a specific task.

Normally, the VBA code is stored in the current document, saved from the VBA Editor when the display is saved, and loaded back to VBA Editor when a display is open in GraphWorX.

However, if the rules described below are followed, the VBA code can be 'bound' to a GraphWorX object. When such an object is pasted/dropped to another instance of GraphWorX, to Symbol Library, or to scrap (desktop), this code goes with it.

How VBA Wizard Works

The VBA Wizard can be run to perform a specific task either in design or runtime mode.

Design Mode

You can launch the VBA Wizard macro in design mode in GraphWorX by double-clicking the VBA Wizard.

By default, when an object is double-clicked in design mode, a Property Inspector is launched. However, if there is a special keyword in the first line of the **Custom Data** field of Property Inspector, a macro can be run.

The format of the keyword for a macro called "MacroName" is:

OnDoubleClick=<GwxMacroName_Main.MacroName>, Parameters=<>

There must exist a macro "MacroName" in the module "GwxMacroName_Main" in VBA to successfully run the macro. The user is allowed to put any string between angle brackets of a "Parameters=<>" section. Any data worth sending to the macro would be put between the angle brackets. These data are then available when the macro runs.

Note: The name of a macro cannot contain spaces.

Runtime Mode

You can launch the VBA Wizard macro in runtime mode by clicking a GraphWorX button or a pick action configured to run a macro. The **Action** field in the **Pick** tab of the Property Inspector must be configured to **Run Script**. The **Script Name** field must contain a macro name in the following format:

GwxMacroName_Main.MacroName

Select **VBA Script** from the drop-down list under **Script Type**. Click the Create button to run the VBA Script Wizard.

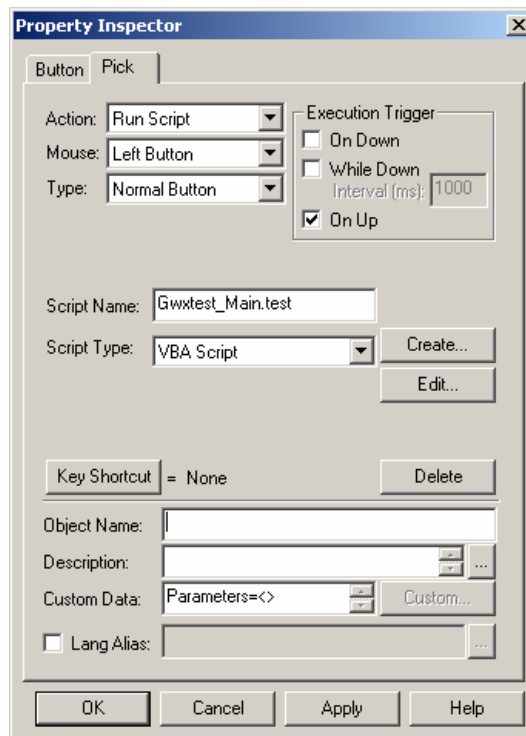


Figure 11.1. VBA Wizard in Pick Action

The **Custom Data** field can contain any string keeping custom data as desired between angle brackets.

Rules for VBA Wizard

GraphWorX takes care of coding behind the VBA Wizard. Due to the special naming convention of all modules belonging to a specific VBA Wizard, the code in these modules can be properly moved with the symbol.

If the macro name of the VBA Wizard is "MacroName," then all code modules must start with the string "GwxMacroName_." This technique allows more code and form modules to be used for one VBA Wizard object, and it facilitates moving all of this code with the object when necessary.

VBA Wizard Creation Tool

Because creating VBA Wizard objects can be a tedious task, GraphWorX offers a **VBA Script Wizard** that converts objects to a VBA Wizard and generates the VBA template code. The code can then be easily enhanced and modified. The following example demonstrates the use of this tool:

1. Open a GraphWorX display.
2. Create several ellipses and group them into a symbol.
3. Right-click on this symbol and select **Create VBA Script** from the popup menu. (You can also select **Macros > Create** from the **Tools** menu.) This opens the **VBA Script Wizard** dialog box, as shown below.
4. Enter a macro name (for example, "Test") in the **Script Name** field. Note, that the **Module** field is unavailable, but it is filled in automatically based on the name typed in the **Script Name** field.

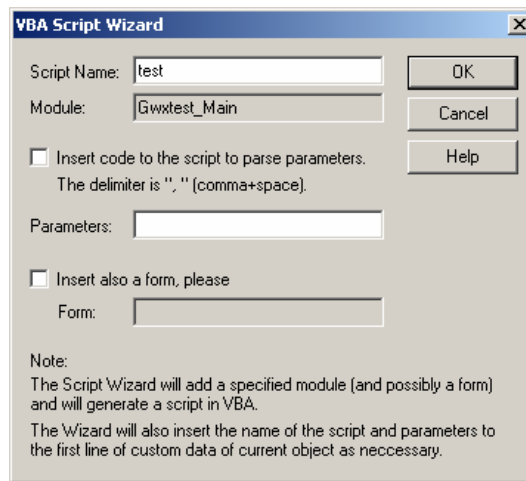


Figure 11.2. VBA Script Wizard Dialog

5. Click **OK**. The Visual Basic Editor opens, as shown below, and the cursor should be placed in the body of the "Test" subroutine in a module "GwxTest_Main." You can now type the code, which is run when you double-click on the symbol in design mode. Try the example code shown in the figure below.

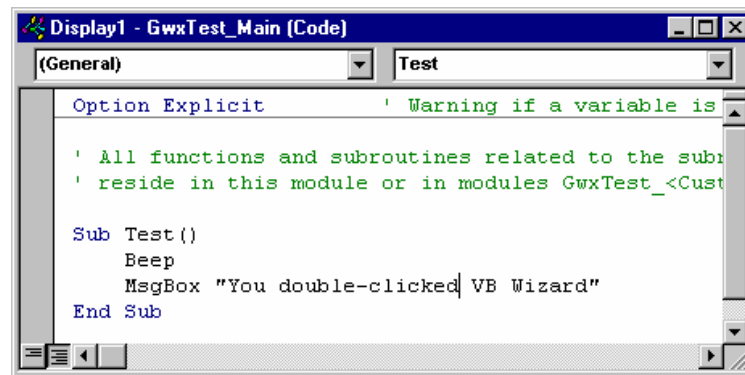


Figure 11.3. Test Subroutine (Generated by Wizard)

6. Return to GraphWorX, and double-click the symbol. A beep should sound, and a message should appear.

You can try other examples with different settings in the following fields of the **VBA Script Wizard** dialog box.

Macro or Script Name

In the **Script Name** field of the **VBA Script Wizard** dialog box, type the name of your macro. The macro name should begin with a character and should contain alphanumeric characters only. If there is already a macro or a module of that name, you must choose another name.

Module

The **Module** field of the **VBA Script Wizard** dialog box is always grayed out because the module name is generated automatically based on the Script Name.

If the **Insert code to the script to parse parameters** check box is checked, some extra code is generated in the body of the macro subroutine. It helps in retrieving and storing parameters from the VBA Wizard object. This code uses the "GwxTools" module to convert your parameters to a string, called "StrPar," that is local to your macro's subroutine.

Parameters

Type any string you like in the **Parameters** field of the **VBA Script Wizard** dialog box. You can obtain this string when the macro runs. This optional field is designed to allow custom data specific to a VBA Wizard instance. Different instances of the same objects can keep different data. This feature serves to simplify your code by allowing you to pass values into a macro.

Insert Also a Form, Please

Check the **Insert also a form, please** check box in the **VBA Script Wizard** dialog box if you need a VBA form to be launched from the macro. You are allowed to create any number of forms for the VBA Wizard, assuming you follow the naming convention (If you do not follow the convention, the VBA code is not moved with the object when necessary.)

Form

The **Form** field in the **VBA Script Wizard** dialog box is always grayed out and is generated automatically based on the Script Name.

More Information About VBA Wizards

Design Mode VBA Wizard

The **Macros** submenu of the **Tools** menu contains several other items commands for VBA Wizards in design mode, as shown in the figure below:

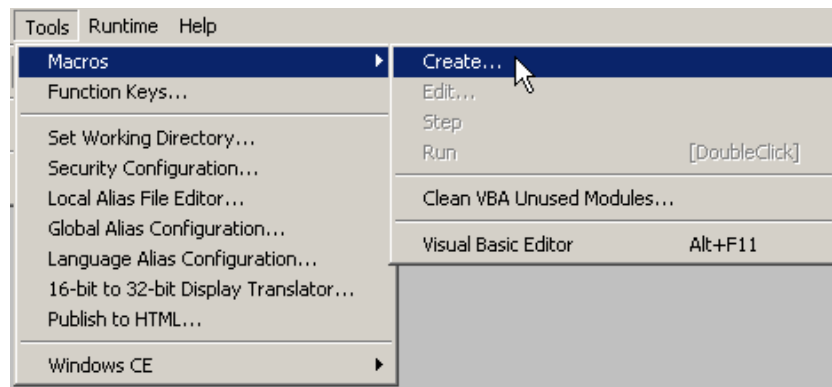


Figure 11.4. VBA Wizard Menus

Create Macro

Selecting **Macros > Create** from the **Tools** menu opens the **VBA Script Wizard** dialog box. You can also right-click on an object and then select **Create VBA Script** from the popup menu. Both menus are enabled if an object that is not VBA Wizard is selected in the display.

Edit Macro

Selecting **Macros > Edit** from the **Tools** menu opens the VBA Editor and places the cursor in the body of the macro. You can also right-click on an object and then select **Edit VBA Script** from the popup menu. Both menus are enabled if a VBA Wizard object is selected in the display.

Step Macro

Selecting **Macros > Step** from the **Tools** menu runs the macro in step (debug) mode. It opens the VBA Editor and stops on the first line of the macro. It is enabled if a VBA Wizard object is selected in the display.

Run Macro

Selecting **Macros > Run** from the **Tools** menu runs the macro. You can also double-click a VBA Wizard. It is enabled if a VBA Wizard object is selected in the display.

Runtime and Design Mode VBA Wizard

Clean VBA unused modules

When a VBA Wizard is deleted or moved out of the current display, the VBA code is not deleted automatically. However, selecting **Macros > Clean VBA Unused Modules** from the **Tools** menu removes all modules from the VBA Editor that start with the "Gwx" string and are not referenced from the currently displayed VBA Wizards (either design or runtime mode based).

Runtime Mode VBA Wizard

If you want to create a new runtime-based VBA Wizard, create a button or a pick action, select **Run Script** in the **Action** field, as shown in the figure below. Select **VBA Script** from the drop-down list under **Script Type**. Click the **Create** button to run the VBA Script Wizard.

Note: You do not need to use the **Edit** command to edit the macro. You can easily open the VBA Editor, find the module you want, and edit the macro directly.

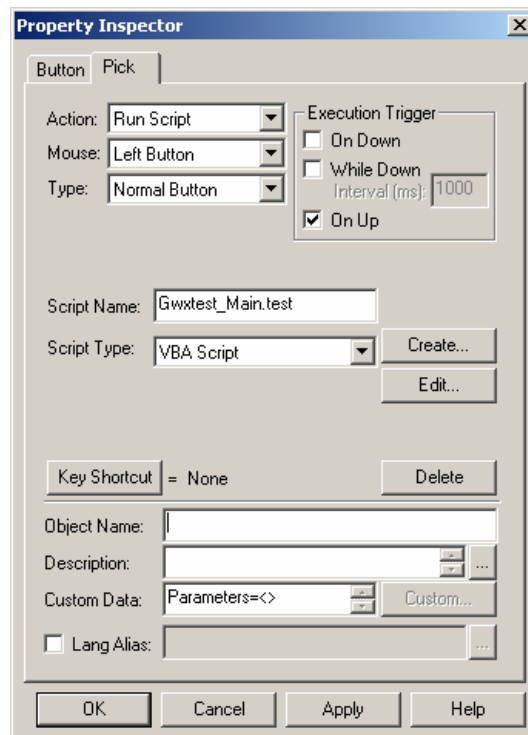


Figure 11.5. VBA Wizard in Pick Action

Other Sources of Information

There are a lot of VBA Wizard objects packaged with GraphWorX. You can find them either in sample displays, or in the Symbol Library.

GraphWorX ActiveX Control

The GraphWorX ActiveX ("GWXview32.ocx") is an ActiveX control capable of running GraphWorX displays. The advantage of ActiveX controls like GraphWorX is that they can be embedded into any control container (e.g. Visual Basic Forms, Internet Explorer HTML pages, GraphWorX displays, etc.).

GraphWorX is essentially a runtime-only component (with some minimal configuration capability); the runtime-only design allows the ActiveX to be compact with respect to memory usage. The GraphWorX ActiveX executes displays created by GraphWorX. GraphWorX has all the runtime capabilities of GraphWorX32.exe, except the ability to execute VBA scripts.

Inserting the GraphWorX ActiveX

Techniques for inserting an ActiveX control may vary slightly among different control containers, however the basics are the same. This section describes how to insert the GraphWorX ActiveX into GraphWorX32.exe.

1. From the **Edit** menu, choose **Insert New Object**, or click the **Insert ActiveX Control/OLE Object** button on the **ActiveX** toolbar, as shown below.



Figure 12.1. Insert ActiveX Control/OLE Object Toolbar Button

2. This opens the **Insert Object** dialog box, shown in the figure below:

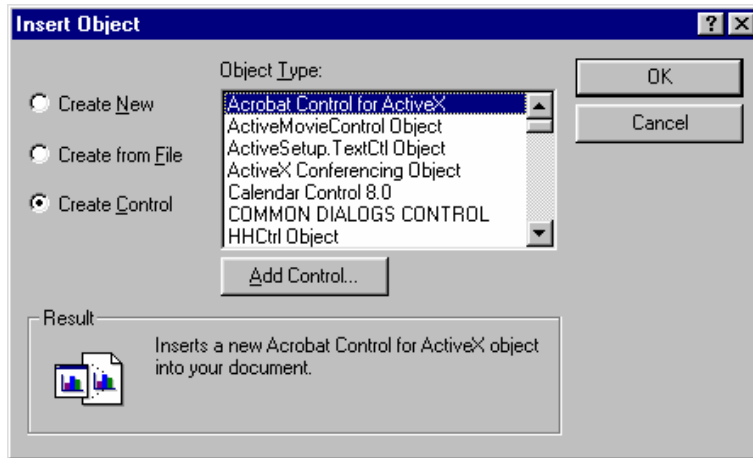


Figure 12.2. Insert Object Dialog Box

3. Select the item **SmAr GWX View ActiveX** from the list of available controls, and then click **OK**. The GraphWorX ActiveX control appears in the center of the display. Alternatively, you can use the **GWXView ActiveX** button on the **ActiveX** toolbar to directly insert the GraphWorX ActiveX.

GraphWorX ActiveX Toolbar

The GraphWorX ActiveX toolbar, which is part of the GraphWorX Screen, is shown below:



Figure 12.3. GraphWorX ActiveX Toolbar

Customizing the ActiveX Toolbar

Any ActiveX that is inserted in the display via the **OLE** button on the ActiveX toolbar is automatically added to the toolbar as a new button. You can remove ActiveX buttons from the toolbar by right-clicking on the toolbar. This opens the **Customize Toolbar** dialog box. Select the ActiveX from the list on the right, and then click **Remove**. Alternatively, you can remove a button from the toolbar by clicking the button on the toolbar while pressing the shift key and dragging the button off the toolbar.

If you want a certain set of ActiveXs to be available after installation, you can use the following registry settings to define a starting set of ActiveXs for the toolbar.

HKEY_LOCAL_MACHINE\Software\Iconics\Gwx32\OEM Information\ActiveX Toolbar Default Info\Number of ProgIDs

This is the number of ActiveX buttons that will appear on the toolbar.

HKEY_LOCAL_MACHINE\Software\Iconics\Gwx32\OEM Information\ActiveX Toolbar Default Info\ProgID_1

HKEY_LOCAL_MACHINE\Software\Iconics\Gwx32\OEM Information\ActiveX Toolbar Default Info\ProgID_2

HKEY_LOCAL_MACHINE\Software\Iconics\Gwx32\OEM Information\ActiveX Toolbar Default Info\ProgID_3

...etc.

These are Programmatic IDs of the ActiveXs to appear on the toolbar (for example, "GWXVIEW32.GWXview32Ctrl.1").

Note: These registry entries will not do anything if GraphWorX has already been run (because the currently saved toolbar settings override these initial settings). If GraphWorX has already been run, you would need to delete the following registry entry:

HKEY_CURRENT_USER\Software\Iconics\Gwx32\Custom Toolbar Settings.

OLE ActiveX



Clicking the **OLE** button on the **ActiveX** toolbar inserts an ActiveX Control/OLE Object. See GraphWorX ActiveX Control - OLE Automation topic for more information.

Graphics ActiveX



Clicking the **GWXView ActiveX** button on the **ActiveX** toolbar opens the Graphics ActiveX.

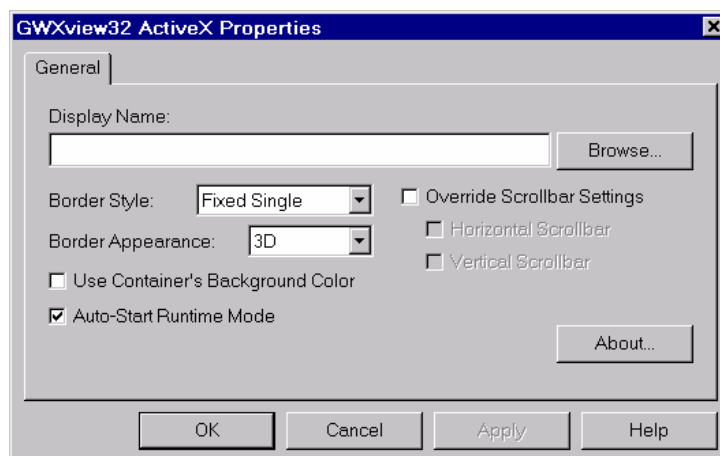


Figure 12.4. GWXView ActiveX Properties Dialog Box

The dialog box shown above allows you to insert a GraphworX Display within this display.

You can save this inserted display with or without VBA. The default is that it is saved with VBA. However, if the display is embedded, it should be saved without VBA as the following error message will display if you attempt to save it with VBA.

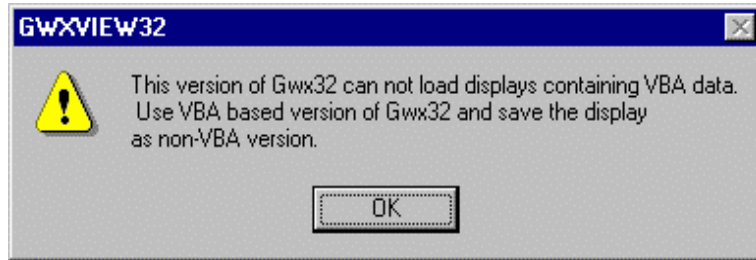


Figure 12.5. Error Message

Trend ActiveX



Clicking the **TWXView** button on the **ActiveX** toolbar opens the TrendWorX Viewer ActiveX. For more information, refer to the TrendWorX Viewer ActiveX help documentation.

Alarm ActiveX



Clicking the **AWXView** button on the **ActiveX** toolbar opens the AlarmWorX Container. For more information, refer to the AlarmWorX Viewer ActiveX help documentation.

Gauge ActiveX



Clicking the **GWXGauge ActiveX** button on the **ActiveX** toolbar opens the Gauge ActiveX, shown below.

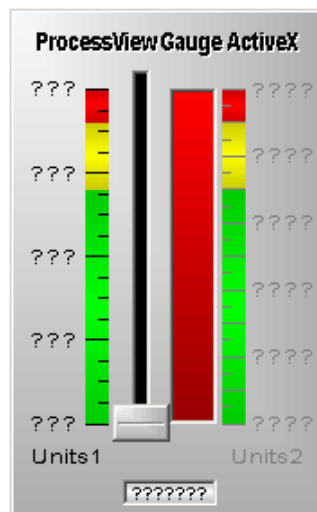


Figure 12.6. Smar Gauge

For more information about the Gauge ActiveX, refer to the **GWXGauge ActiveX** help documentation.

Switch ActiveX



Clicking the **GWXSwitch ActiveX** button on the **ActiveX** toolbar opens the Switch ActiveX, shown below.



Figure 12.7. Smar Switch

For more information about the Switch ActiveX, refer to the **GWXSwitch ActiveX** help documentation.

Slider ActiveX



Clicking the **GWXSlider ActiveX** button on the **ActiveX** toolbar opens the Slider ActiveX, shown below.

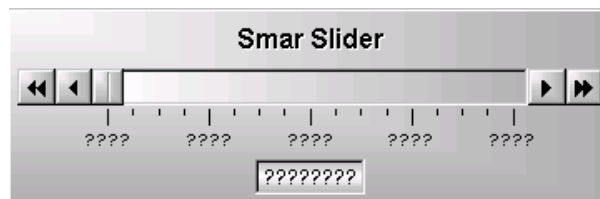


Figure 12.8. Slider ActiveX

For more information about the Slider ActiveX, refer to the **GWXSlider ActiveX** help documentation.

Numeric ActiveX



Clicking the **GWXNumeric ActiveX** button on the **ActiveX** toolbar opens the Numeric ActiveX, shown below.

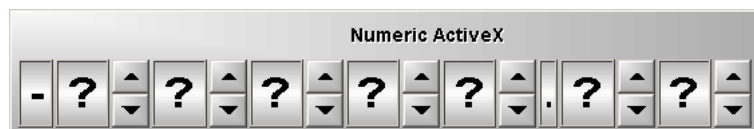


Figure 12.9. Numeric ActiveX

For more information about the Numeric ActiveX, refer to the **GWXNumeric ActiveX** help documentation.

Vessel ActiveX



Clicking the **GWXVessel ActiveX** button on the **ActiveX** toolbar opens the Vessel ActiveX, shown below.

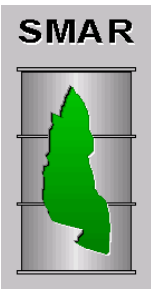


Figure 12.10. Vessel ActiveX

For more information about the Vessel ActiveX, refer to the **GWXVessel ActiveX** help documentation.

National Instruments ActiveX



Clicking the **National Instruments ActiveX Control** button on the **ActiveX** toolbar opens the National Instruments ActiveX Control, shown below.

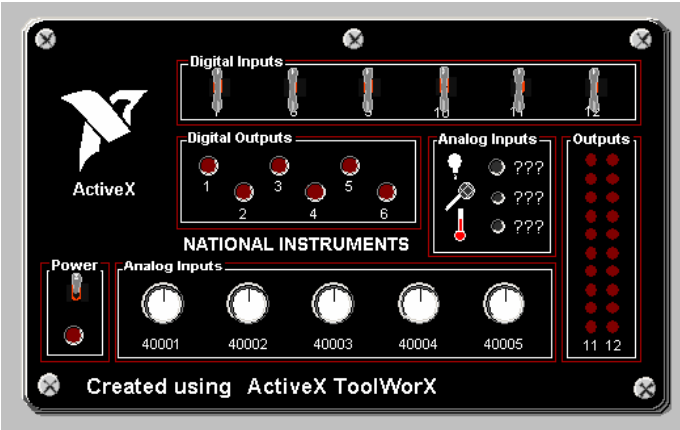


Figure 12.11. National Instruments ActiveX Control

For more information about the National Instruments ActiveX, refer to the **National Instruments ActiveX** help documentation.

Configuring the GraphWorX ActiveX

The **GWXView ActiveX Properties** dialog box, shown below, allows you to change certain attributes of the GraphWorX ActiveX control.

To configure the GraphWorX Active X:

1. Double-click the GraphWorX ActiveX to display the control's **Properties** dialog box, as shown below.
2. Fill in the parameters described in the table below, and then click **OK**.

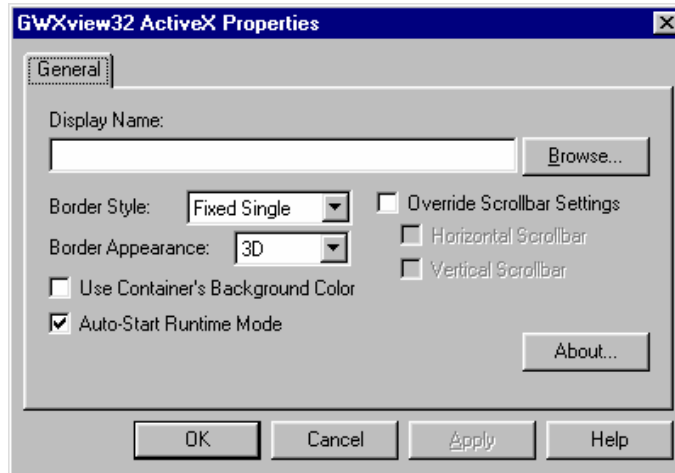


Figure 12.12. GraphWorX Property Page

GraphWorX ActiveX Parameters

Parameter	Description
Display Name	Specifies the file name of the GraphWorX display to be loaded in this control. Click the Browse button to search for display files.
Border Style	Specifies the appearance of the window border: "No Border" or "Fixed Single."
Border Appearance	Specifies the appearance of the window border: flat or three-dimensional.
Use Container's Background Color	When this option is checked, the background color of the control will automatically be set to match the background color of the container in which this control is embedded. This property only works if the container supports the "AmbientBackColor" property. GraphWorX and Visual Basic Forms both support this property.
Auto-Start Runtime Mode	When checked, the control will automatically enter runtime mode when the control's container enters runtime mode. This property only works if the container supports the AmbientUserMode property. Both GraphWorX and Visual Basic Forms support this property.
Override Scrollbar Settings	When this option is checked, this property indicates the control will override the scrollbar visibility settings of the display currently loaded in the control. When it is not checked, the control will use the scrollbar settings defined in the currently loaded display.
Horizontal Scrollbar	When this option is checked, the horizontal scrollbar of the control's window is visible. When it is not checked, the horizontal scrollbar is hidden. This property is ignored if Override Scrollbar Settings is not checked.
Vertical Scrollbar	When this option is checked, the vertical scrollbar of the control's window is visible. When it is not checked, the vertical scrollbar is hidden. This property is ignored if Override Scrollbar Settings is not checked.

OLE Automation Reference

Introduction

OLE automation is used to access properties and methods of objects in GraphWorX.

Properties are used to reference attributes if an object (for example, a GraphWorX display) has a *BackgroundColor* property. Methods are used to make the object perform an action (for example, a display has a *FileOpen* method that loads a new display).

Properties and methods can be called from Visual Basic for Applications (VBA), VBScript, or JavaScript, or from C++ programs.

There are numerous object types in GraphWorX, each with its own methods and properties. This section describes in detail, the methods and properties for the object types exposed by GraphWorX.

Registration at GenRegistrar in Design Mode

GraphWorX also registers at GenRegistrar in design mode. In design mode, GraphWorX registers the dispatch pointer to the display with a keyword "GWX32_DESIGN_MODE".

When entering runtime mode, it re-registers the dispatch pointer to <current-display-title> and keeps this key as long as it is in runtime, regardless of what other displays are loaded. This provides an easy way to recognize miscellaneous instances of GraphWorX in runtime (based on the caption with which they started).

The Design Time Registration feature is by default off, and it can be turned on using the registry key DesignTimeRegistration under GWX32 / Compatibility Properties, as shown in the figure below:

```
0 .. feature disabled
1 .. feature enabled
```

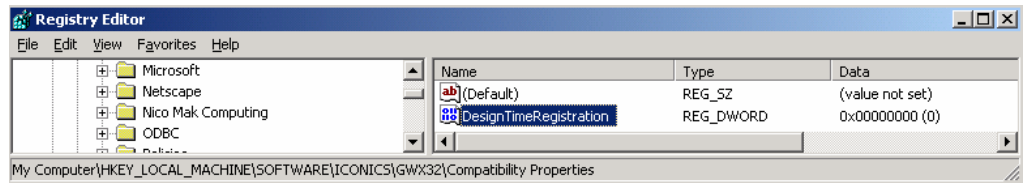


Figure 13.1. Enabling Design Mode Registration at GenRegistrar

When leaving runtime mode, GraphWorX registers again with the key "GWX32_DESIGN_MODE". The following example shows how an instance of GraphWorX in design mode can be obtained from a Visual Basic application:

```
Sub AutomateGwx()
    ' Note: Add SMAR AutoGenRegistrar to Project | References
    ' Note: Add GWX32 to Project | References

    ' Create an Automation GenRegistrar Wrapper
    Dim genreg As New AutoGenRegistrar
    If genreg Is Not Nothing Then Exit Sub ' failure

    ' Declare GWX reference and get it from GenRegistrar
    ' Note that we ask for GWX instance in design mode
    Dim gwx As Gwx32.GwxDisplay
    Call genreg.GetDispatch("GWX32_DESIGN_MODE", "GWX32", "*", gwx)
    If gwx Is Nothing Then Exit Sub ' failure

    ' Automate GWX instance
    gwx.ShowWindow
    gwx.BringWindowToTop
    Randomize
    gwx.BackgroundColor = RGB(Rnd * 255, Rnd * 255, Rnd * 255)
End Sub
```

Object Hierarchy

The following diagram outlines the hierarchy of objects exposed by GraphWorX via OLE Automation.

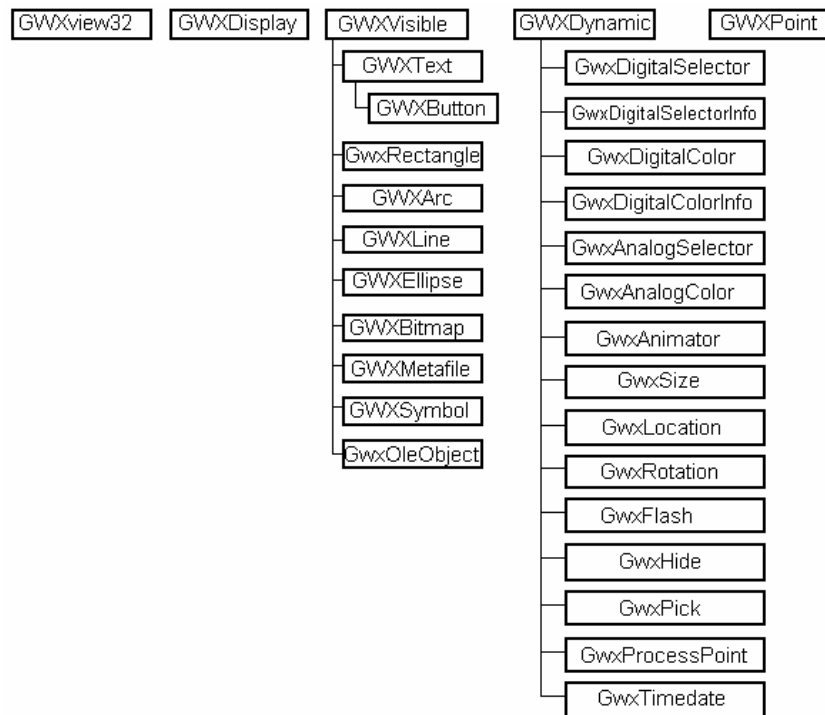


Figure 13.2. OLE Automation Hierarchy

GWXview is the data type of GWXview ActiveX control. For developers using ActiveX ToolWorX, **GWXview** is the data type of the base ActiveX control from which new controls are derived (new controls can inherit the properties and methods of GWXview). Each GWXview object contains a single **GwxDisplay** object.

GwxDisplay is the object type for GraphWorX displays. A GraphWorX display contains collections of **GwxVisible** derived objects, **GwxDynamic** derived objects, and **GwxPoint** objects. The **GwxDisplay** object exposes methods for creating and getting existing instances of these visible and dynamic objects.

GwxVisible is the base object type for objects in a GraphWorX display that can be seen. All visible GraphWorX objects (i.e. **GwxRectangle**, **GwxEllipse**, **GwxText**, etc.) are derived from **GwxVisible** and consequently inherit all the properties and methods of **GwxVisible**.

GwxDynamic is the base object type for objects in a GraphWorX display that perform a dynamic transformation on an associated visible object. For example, a **GwxSize** object associated with a **GwxRectangle** object would change the size of the rectangle based on the OPC data value associated with the **GwxSize** object. All dynamic GraphWorX objects (i.e. **GwxSize**, **GwxLocation**, **GwxRotation**, etc.) are derived from **GwxDynamic** and consequently inherit all the properties and methods of **GwxDynamic**. Dynamic objects are not visible themselves; they exist to act upon the visual properties of an associated **GwxVisible** derived object. A **GwxVisible** object can have many **GwxDynamic** objects associated with it, by a **GwxDynamic** object has only one associated **GwxVisible** object.

GwxPoint is the base object type for objects in a GraphWorX display that reference OPC data, expressions, local variables, or constant values. Each **GwxDynamic** object may have one or more **GwxPoint** objects associated with it. **GwxPoint** objects are created and destroyed by **GwxDynamic** objects as needed.

Automation Programming in C++ and Visual Basic: Understanding Inheritance

The GraphWorX OLE Automation object hierarchy makes extensive use of inheritance. This raises some programming issues in both C++ and Visual Basic. To understand these programming issues, you must first understand how methods and properties are accessed via OLE Automation interfaces.

An OLE Automation method or property can be accessed by the method/property name or by an ID number associated with the method/property. When derived objects use inherited methods/properties from a base object, the ID numbers for the inherited methods/properties in the derived object must be changed such that the high word of the ID number is set to the level of inheritance of the derived object from base object in which the method property is defined. For example, **GwxVisible** defines the **FillColor** property to have an ID number of **0x3**. **GwxRectangle** inherits the **FillColor** property as having the ID number **0x10003** (because **GwxRectangle** is derived from **GwxVisible**). Similarly, **GwxButton** inherits the **FillColor** property as having the ID number **0x20003** (because **GwxButton** is derived from **GwxText** which is derived from **GwxVisible**).

The nature of ID numbering for inheritance does not cause any problems when accessing objects via their actual object types. Programming issues arise when you want to downcast a derived object type to a base object type. For example, the **GwxDisplay** method **GetVisibleObjectFromName** returns a **GwxVisible** derived object, but there may be no way to know what the actual object type is ahead of time, so the programmer would want to assign the return value to a **GwxVisible** object (rather than a specific object type like **GwxRectangle** or **GwxEllipse**). But, if you then try to access methods/properties defined for **GwxVisible** of the returned object, they will not work because the ID numbers will be wrong (if the returned object was actually a **GwxRectangle** the ID for **FillColor** would be **0x10003**, but since the object is being accessed via the base **GwxVisible** object type it will try to use the ID number **0x3**).

In Visual Basic, the solution to this issue is simple. If you do not know the actual type of an object and you want to access it generically, use the object type **Object**. This will cause VB to access methods and properties of the object by name rather than by ID, thereby avoiding ID numbering issues.

In C++ this issue can require more work to get around. Typically when using OLE Automation objects in C++ you will allow Developer Studio to generate some kind of wrapper code for the automation interface. For instance, you can use Class Wizard to generate a **COleDispatchDriver** derived wrapper class. Or, you can use the **#import** directive to generate interface wrapper code (TLI and TLH files). Both these techniques however, access methods and properties via hard-coded ID numbers. Hence, when using such wrapper classes you will not be able to correctly downcast object types. The best way around this issue is to modify the generated wrapper classes for **GwxVisible** and **GwxDynamic** to access methods and properties by name rather than by hard-coded ID.

Events

This section describes the events that are exposed to GraphWorX's integrated Visual Basic for Applications scripting. These events are also exposed from the **GWXview** ActiveX control. (*Note: for developers using ActiveX ToolWorX, there are virtual member functions which correspond to the events described below.*)

Event DataEntryActivated(DataEntry As Object)

This event is fired when the user activates a data entry object (i.e. clicks on a data entry object to type in a new value). The parameter **DataEntry** is the **GwxProcessPoint** object that caused the event.

Event DataEntryDeactivated(DataEntry As Object)

This event is fired when the user deactivates a data entry object (i.e. the user clicks away from a data entry or uses the escape key to cancel editing, or after the user enters a value). The parameter **DataEntry** is the **GwxProcessPoint** object that caused the event.

Event DataEntryValueEntered(DataEntry As Object)

This event is fired when the user enters a new data value into a data entry object. The parameter **DataEntry** is the **GwxProcessPoint** object that caused the event.

Event DialBeginDrag(Dial As Object)

This event is fired when the user clicks the mouse down on a Gwx dial. The parameter **Dial** is the **GwxRotation** object of the dial that caused the event.

Event DialDrag(Dial As Object)

This event is fired when the user drags a Gwx dial (i.e. moving the mouse while the mouse button is held down on the dial). The parameter **Dial** is the **GwxRotation** object of the dial that caused the event.

Event DialEndDrag(Dial As Object)

This event is fired when the user releases the mouse button on a Gwx dial. This event is also called when the user turns the dial with the arrow keys instead of the mouse. The parameter **Dial** is the **GwxRotation** object of the dial that caused the event.

Event DisplayKeyDown(KeyCode As Long, Shift As Long, CancelProcessing)

Event DisplayKeyUp(KeyCode As Long, Shift As Long, CancelProcessing)

The above events are fired when a key is pressed (DisplayKeyDown) or released (DisplayKeyUp). It is possible to suppress the processing of the events by GraphWorX by setting CancelProcessing to True.

Note: This event will be fired only in the case it is enabled either programmatically (EnableRuntimeEvents) or manually in the **Format > Application Preferences > VBA** tab.

Parameters:

KeyCode

Key code, such as **vbKeyHome** (the HOME key).

Shift

Integer that corresponds to the state of the SHIFT, CTRL, and ALT keys at the time of the event.

CancelProcessing

Variant that can suppress further processing of the event by GraphWorX if set to True.

The following table shows the values for the *Shift* constants.

<i>Constant</i>	<i>Value</i>	<i>Description</i>
vbShiftMask	1	SHIFT key bit mask.
vbCtrlMask	2	CTRL key bit mask.
vbAltMask	4	ALT key bit mask.

Event DisplayLoad()

This event is fired just after a display is loaded into GraphWorX.

Event DisplayMouseDown(Button As Long, Shift As Long, X As Single, Y As Single, CancelProcessing)

Event DisplayMouseMove(Button As Long, Shift As Long, X As Single, Y As Single, CancelProcessing)

Event DisplayMouseUp(Button As Long, Shift As Long, X As Single, Y As Single, CancelProcessing)

The above events are fired when a mouse button is double-clicked (DisplayMouseDown), pressed (DisplayMouseMove), or released (DisplayMouseUp), or the mouse position is changed (DisplayMouseUp). It is possible to suppress the processing of the event by GraphWorX by setting CancelProcessing to True.

Note: This event will be fired only in the case it is enabled either programmatically (EnableRuntimeEvents) or manually in the **Format > Application Preferences > VBA** tab.

Parameters:

button

An integer that corresponds to the state of the mouse buttons in which a bit is set if the button is down. The *button* argument is a bit field with bits corresponding to the left button (bit 0), right button (bit 1), and middle button (bit 2). These bits correspond to the values 1, 2, and 4, respectively. It indicates the complete state of the mouse buttons; some, all, or none of these three bits can be set, indicating that some, all, or none of the buttons are pressed.

Shift

An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys. A bit is set if the key is down. The *shift* argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. The *shift* argument indicates the state of these keys. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT were pressed, the value of *shift* would be 6.

x, y

A number that specifies the current location of the mouse pointer. The *x* and *y* values are always expressed in terms of the coordinate system set by the **ScaleHeight**, **ScaleWidth**, **ScaleLeft**, and **ScaleTop** properties of the object.

CancelProcessing

Variant that can suppress further processing of the event by GraphWorX if set to True.

Event DisplayUnload()

This event is fired just before a display is unloaded from GraphWorX (i.e. just prior to loading a new display into GraphWorX or prior to closing GraphWorX).

void LanguageChanged(long LocaleID)

This event is fired when GraphWorX is switched to a new language. The parameter **localeID** is the locale ID of the new language. (*Not available for displays running in GWXview objects*).

Event PickPostDown(Pick As Object)

This event is fired when the user presses down on a Gwx pick action. The event is fired after the pick's "OnDown" action is executed. The parameter **Pick** is **GwxPick** object of the Pick dynamic that caused the event.

Event PickPostUp(Pick As Object)

This event is fired when the user releases (mouse button up) a Gwx pick action. The event is fired after the pick's "OnUp" action is executed. The parameter **Pick** is **GwxPick** object of the Pick dynamic that caused the event.

Event PickPostWhileDown(Pick As Object)

This event is fired while the pick action is held down at time interval specified in the pick object. The event is fired after the pick's "WhileDown" action is executed. The parameter **Pick** is **GwxPick** object of the Pick dynamic that caused the event.

Event PickPreDown(Pick As Object)

This event is fired when the user presses down on a Gwx pick action. The event is fired before the pick's "OnDown" action is executed. The parameter **Pick** is **GwxPick** object of the Pick dynamic that caused the event.

Event PickPreUp(Pick As Object)

This event is fired when the user releases (mouse button up) a Gwx pick action. The event is fired before the pick's "OnUp" action is executed. The parameter **Pick** is **GwxPick** object of the Pick dynamic that caused the event.

Event PickPreWhileDown(Pick As Object)

This event is fired while the pick action is held down at time interval specified in the pick object. The event is fired before the pick's "WhileDown" action is executed. The parameter **Pick** is **GwxPick** object of the Pick dynamic that caused the event.

Event PostAnimateDisplay()

This event is fired after GraphWorX has finished animating a display.

Event PostAnimateLayer(LayerName As String)

This event is fired after GraphWorX has finished animating the specified layer.

Event PostDeanimateDisplay()

This event is fired after GraphWorX has finished deanimating a display.

Event PostDeanimateLayer(LayerName As String)

This event is fired after GraphWorX has finished deanimating the specified layer.

Event PostRuntimeStart()

This event is fired when GraphWorX has finished entering runtime mode.

Event PostRuntimeStop()

This event is fired after GraphWorX has completely exited runtime mode.

void PreAnimateDisplay()

This event is fired just before GraphWorX animates a display (animating a display is the process of requesting the tags from the OPC server(s)). Displays are animated when loaded during runtime mode, and deanimated when unloaded during runtime mode.

Event PreAnimateDisplay()

This event is fired just before GraphWorX animates the specified display (animating a display is the process of requesting the tags from the OPC server(s) that are referenced in the display).

Event PreAnimateLayer(LayerName As String)

This event is fired just before GraphWorX animates the specified display layer (animating a layer is the process of requesting the tags from the OPC server(s) that are referenced in the layer).

Event PreDeanimateDisplay()

This event is fired just before GraphWorX deanimates a display (deanimating a display is the process of releasing the tags previously requested from the OPC server(s)).

Event PreDeanimateLayer(LayerName As String)

This event is fired just before GraphWorX deanimates the specified display layer (deanimating a layer is the process of releasing the tags previously requested from the OPC server(s)).

Event PreRuntimeStart()

This event is fired just before GraphWorX is put into runtime mode.

Event PreRuntimeStop()

This event is fired just before GraphWorX exits runtime mode.

Event SliderBeginDrag(Slider As Object)

This event is fired when the user clicks the mouse down on a Gwx slider. The parameter **Slider** is the **GwxLocation** object of the slider that caused the event.

Event SliderDrag(Slider As Object)

This event is fired when the user drags a Gwx slider (i.e. moving the mouse while the mouse button is held down on the slider). The parameter **Slider** is the **GwxLocation** object of the slider that caused the event.

Event SliderEndDrag(Slider As Object)

This event is fired when the user releases the mouse button on a Gwx slider. This event is also called when the user moves the slider with the arrow keys instead of the mouse. The parameter **Slider** is the **GwxLocation** object of the slider that caused the event.

Properties and Methods

GraphWorX includes the following general OLE Automation classes:

- **GwxView**
- **GwxDisplay**
- **GwxVisible**
- **GwxDynamic**
- **GwxPoint**

GWXview

GWXview is the object type for the GraphWorX ActiveX control (GWXview32.ocx). When using ActiveX ToolWorX, the properties and methods described in this section can be inherited by new ActiveX controls. **GWXview** objects can be created via the ProgID **GWXVIEW32.GWXview32Ctrl.1**.

Properties

AutoStartRuntime As Boolean

When TRUE, the object/control will automatically enter runtime mode when the object's container enters runtime mode. This property only works if the container supports the AmbientUserMode property.

HorizontalScrollbar As Boolean

When TRUE, the horizontal scrollbar of the control's window is visible, when FALSE the horizontal scrollbar is hidden. This property is ignored if **OverrideScrollbarSettings** is FALSE.

OverrideScrollbarSettings As Boolean

When TRUE, this property indicates the object/control will override the scrollbar visibility settings of the display currently loaded in the control. When FALSE, the object/control will use the scrollbar settings defined in the currently loaded display.

UseAmbientBackColor As Boolean

When TRUE, the background color of the object/control will automatically be set to match the background color of the container in which this object is embedded. This property only works if the container supports the AmbientBackColor property.

VerticalScrollbar As Boolean

When TRUE, the vertical scrollbar of the control's window is visible, when FALSE the vertical scrollbar is hidden. This property is ignored if **OverrideScrollbarSettings** is FALSE.

DisplayName As String

File name of the GraphWorX display loaded in this object/control. Setting this property to a new file name will cause the new display to be loaded.

Appearance As Integer

Appearance of the window border. Valid values are:

Flat	=	0
3-D	=	1

BorderStyle As Integer

Border style of the window. Valid values are:

No border	=	0
Normal Border	=	1

Methods

Function GetDisplay() As Object

Returns the GwxDisplay object for the display currently loaded in this object/control. You can then use this object to access all of the properties and methods of all the object types described in this section.

Function ReplaceFilePath(OldSubstring As String, NewSubstring As String) As Long

Works over path name parameters of pick dynamics. Returns 0 on success and HRESULT when something fails.

```
'similar to ReplaceTag
'VBA example, works only in configure mode
'replaces file path in whole display

Dim Status As Long
Status = ThisDisplay.ReplaceFilePath("C:\Windows\Temp", "D:\Temp")
'if Status <> 0 then there was no replacement performed or an error
occurs
If Status <> 0 Then
    MsgBox "No replacements"
Else
    MsgBox "Tags replaced"
End If
```

Function ReplaceHost(OldHostName As String, NewHostName As String) As Long

Works over both data source tags and path name attributes and replaces node name substring within URL path only. Returns 0 on success and HRESULT when something fails.

```
'similar to ReplaceTag
'VBA example, works only in configure mode
'replaces host name in whole display

Dim Status As Long
Status = ThisDisplay.ReplaceHost("Host1", "Host2")
'if Status <> 0 then there was no replacement performed or an error
occurs
If Status <> 0 Then
    MsgBox "No replacements"
Else
    MsgBox "Tags replaced"
End If
```

Function ReplaceHostEx(OldHostNameSubstring As String, NewHostNameSubstring As String, MatchCase As Boolean, MatchWholeWord As Boolean) As Long

Works over both data source tags and path name attributes and replaces node name substring within URL path only, and supports case-sensitivity, wildcard strings and MatchWholeWord flag. Returns 0 on success and HRESULT when something fails.

```
'similar to ReplaceTag
'VBA example, works only in configure mode
'replaces host name in whole display, regarding "case" and "whole
words" options

Dim Status As Long
Status = ThisDisplay.ReplaceHostEx("Host1", "Host2", True, True)
'if Status <> 0 then there was no replacement performed or an error
occurs
If Status <> 0 Then
    MsgBox "No replacements"
Else
    MsgBox "Tags replaced"
```


End If

Function ReplaceTag(OldSubstring As String, NewSubstring As String) As Long

Works over data source tags in dynamic actions of all display objects. Replaces OldSubstring with NewSubstring in certain situations. Returns 0 on success and HRESULT when something fails.

Typical use of this method:

```
'VBA example, works only in configure mode
'replaces tag name in whole display

Dim Status As Long
Status = ThisDisplay.ReplaceTag("gfwsim.ramp.float",
"gfwsim.sine.double")
'if Status <> 0 then there was no replacement performed or an error
occurs
If Status <> 0 Then
  MsgBox "No replacements"
Else
  MsgBox "Tags replaced"
End If
```

long ReplaceStringInString (BSTR StringToReplace, BSTR OldSubstring, BSTR NewSubstring)

Replaces OldSubstring with NewSubstring inside (within) StringToReplace. Returns 0 on success and HRESULT when something fails.

Sub SelectLanguage(Language As String, LanguageSubset As String, LocaleID As Long, SynchronizeLCID As Boolean)

Sub SetGasParentCookie(newCookie As Long)

Used internally for communication between ActiveX control and container. Do not modify.

Sub StartRuntime()

Puts this object/control into runtime mode.

Sub StopRuntime()

Takes this object/control out of runtime mode.

GwxDisplay

GwxDisplay is the object type for GraphWorX displays. Properties and methods of the **GwxDisplay** object type are described in detail below. Also described, are the events exposed to GraphWorX's integrated Visual Basic for Applications (VBA). **GwxDisplay** objects can be created via the ProgID **Gwx.Display**.

Properties

BackgroundColor As OLE_COLOR

Gets/Sets the current display's background color.

Redraw As Boolean

When TRUE, automation calls when change the visual appearance of a display will automatically refresh the display. When FALSE, the user must explicitly refresh the display (for instance, you may want to change the attributes of many objects, and refresh them all at once).

ScaleMode As GWXSCALEMODE

Sets the scaling mode of the current display. Valid values are:

FixedScale = 0

Scaleable = 1

ScaleablePreserveAspect = 2

Methods

Sub BringWindowToTop()

Brings the GraphWorX main window to the top of the window z-order. *(Not available for displays running in GWXview objects).*

Sub ClearGradientFill()

Clears a previously set background gradient fill (restores solid background fill).

Sub CloseAllPopupWindows()

Closes all currently open popup windows. *(Not available in ActiveX ToolWorX).*

Function ClosePopupWithTitle(titleSubstring As String) As Boolean

Closes the popup window that has the specified substring as part or all of the text in the popup window's title bar. Returns TRUE if the specified popup window was successfully closed, FALSE otherwise. *(Not available in ActiveX ToolWorX).*

Sub CloseWindow()

Closes the window for this display (this method is like the **ExitApplication** method, however **CloseWindow** also works for displays running in popup windows and embedded popup windows). *(Not available in ActiveX ToolWorX).*

Function ConvertGwxSymbolToLibraryObject(gwxSymbol As Object, libraryObject As Object) As Boolean

Converts GwxSymbol object into Symbol Library object. As such can be inserted into one of Symbol Library category files (.sdf) via Symbol Library automation. (See the Symbol Library Help documentation for more information.)

Note: This function works in GraphWorX design mode only.

Function CreateAnalogColorDynamic(visibleObjectName As String, dynamicObjectName As String, changeFill As Boolean, changeLine As Boolean, changeShadow As Boolean, startFillColor As OLE_COLOR, endFillColor As OLE_COLOR, startLineColor As OLE_COLOR, endLineColor As OLE_COLOR, startShadowColor As OLE_COLOR, endShadowColor As OLE_COLOR, defaultColorAbove As Boolean, defaultColorBelow As Boolean) As Object

Creates a **GwxAnalogColor** object with the specified attributes and attaches it to the visible object with the specified object name. Returns the newly created **GwxAnalogColor** object on success, NULL if the operation failed. (See also **GwxAnalogColor** and **GwxDynamic**). *Available only in configure mode.*

Function CreateAnalogSelector(symbolName As String, dynamicObjectName As String, hiddenWhenAbove As Boolean, hiddenWhenBelow As Boolean) As Object

Creates a **GwxAnalogSelector** object with the specified attributes and attaches it to the **GwxSymbol** object with the specified object name. Returns the newly created **GwxAnalogSelector** object on success, NULL if the operation failed. (See also **GwxAnalogSelector**, **GwxSymbol**, **GwxVisible**, and **GwxDynamic**). *Available only in configure mode.*

Function CreateAnimator(symbolName As String, dynamicObjectName As String, animateWhenTrue As Boolean, visibleWhenOff As Boolean, currentFrameWhenOff As Boolean) As Object

Creates a **GwxAnimator** object with the specified attributes and attaches it to the **GwxSymbol** object with the specified object name. Returns the newly created **GwxAnimator** object on success, NULL if the operation failed. (See also **GwxAnimator**, **GwxSymbol**, **GwxVisible**, and **GwxDynamic**). *Available only in configure mode.*

Function CreateArc(arcType As GWXARCTYPE, centerX As Single, centerY As Single, radiusX As Single, radiusY As Single, startAngle As Single, endAngle As Single, isFilled As Boolean, fillColor As OLE_COLOR, lineColor As OLE_COLOR, lineWidth As Long, lineStyle As GWXLINestyle, hasShadow As Boolean, shadowColor As OLE_COLOR, edgeStyle As GWX3DEDGESTYLE, isHidden As Boolean, objectName As String) As Object

Creates and returns a **GwxArc** object with the specified attributes, in the current display. Angles are specified in degrees. (See also **GwxArc** and **GwxVisible**). *Available only in configure mode.*

Function CreateBitmap(filename As String, left As Single, top As Single, width As Single, height As Single, isTransparent As Boolean, transparentColor As Long, lineColor As Long, lineWidth As Long, lineStyle As GWXLINESTYLE, hasShadow As Boolean, shadowColor As Long, edgeStyle As GWX3DEDGESTYLE, isHidden As Boolean, objectName As String) As Object

Creates and returns a **GwxBitmap** object with the specified attributes, in the current display. (See also **GwxVisible**). The **fileName** parameter refers to the file name of a BMP file. *Available only in configure mode.*

Function CreateButton(buttonType As GWXBUTTONTYPE, x As Single, y As Single, label As String, alignment As GWXTEXTALIGNMENT, stretchText As Boolean, isFilled As Boolean, fillColor As OLE_COLOR, lineColor As OLE_COLOR, lineWidth As Long, lineStyle As Long, hasShadow As Boolean, shadowColor As OLE_COLOR, edgeStyle As Long, isHidden As Boolean, objectName As String) As Object

Creates and returns a **GwxButton** object with the specified attributes, in the current display. A button object will be non-operational until a **GwxPick** object is attached to it. (See also **GwxButton**, **GwxPick**, **GwxDynamic**, **GwxText**, and **GwxVisible**). *Available only in configure mode.*

Function CreateDigitalColorDynamic(visibleObjectName As String, dynamicObjectName As String, changeColorWhenTrue As Boolean, changeFill As Boolean, changeLine As Boolean, changeShadow As Boolean, fillColor As OLE_COLOR, lineColor As OLE_COLOR, shadowColor As OLE_COLOR, dataSource As String) As Object

Creates a **GwxDigitalColor** object with the specified attributes and attaches it to the visible object with the specified object name. The object is created with one initial data connection. Additional data connections can be added by calling this function again for the same visible object. Returns the newly created **GwxDigitalColor** object on success, NULL if the operation failed. (See also **GwxDigitalColor**, **GwxDigitalColorInfo**, and **GwxDynamic**). *Available only in configure mode.*

Function CreateDigitalSelector(symbolName As String, dynamicObjectName As String) As Object

Creates a **GwxDigitalSelector** object with the specified attributes and attaches it to the **GwxSymbol** object with the specified object name. Returns the newly created **GwxDigitalSelector** object on success, NULL if the operation failed. The newly created object has no data connections; to add data connections, use **GwxDigitalSelector::SetConnectionInfo**. (See also **GwxDigitalSelector**, **GwxDigitalSelectorInfo**, **GwxSymbol**, **GwxVisible**, and **GwxDynamic**). *Available only in configure mode.*

Function CreateEllipse(left As Single, top As Single, width As Single, height As Single, isFilled As Boolean, fillColor As OLE_COLOR, lineColor As OLE_COLOR, lineWidth As Long, lineStyle As GWXLINESTYLE, hasShadow As Boolean, shadowColor As OLE_COLOR, edgeStyle As GWX3DEDGESTYLE, isHidden As Boolean, objectName As String) As Object

Creates and returns **GwxEllipse** object with the specified attributes, in the current display. (See also **GwxEllipse** and **GwxVisible**). *Available only in configure mode.*

Function CreateFlashDynamic(visibleObjectName As String, dynamicObjectName As String, hideObject As Boolean, flashWhenTrue As Boolean, altStateWhenOff As Boolean, changeFill As Boolean, changeLine As Boolean, changeShadow As Boolean, altFillColor As OLE_COLOR, altLineColor As OLE_COLOR, altShadowColor As OLE_COLOR) As Object

Creates a **GwxFlash** object with the specified attributes and attaches it to the visible object with the specified object name. Returns the newly created **GwxFlash** object on success, NULL if the operation failed. (See also **GwxFlash** and **GwxDynamic**). *Available only in configure mode.*

Function CreateHideDynamic(visibleObjectName As String, dynamicObjectName As String, hideWhenTrue As Boolean, disableObject As Boolean) As Object

Creates a **GwxHide** object with the specified attributes and attaches it to the visible object with the specified object name. Returns the newly created **GwxHide** object on success, NULL if the operation failed. (See also **GwxHide** and **GwxDynamic**). *Available only in configure mode.*

Sub CreateLayer(layerName As String)

Creates a new Layer in the display and assigns to it the name specified in the parameter.

Function CreateLocationDynamic(visibleObjectName As String, dynamicObjectName As String, offsetX As Single, offsetY As Single, slider As Boolean, tracking As Boolean, numberOfDetents As Integer, continuousUpdate As Boolean) As Object

Creates a **GwxLocation** object with the specified attributes and attaches it to the visible object with the specified object name. The parameters **offsetX** and **offsetY** refer to the distance the object will travel from its current location. Returns the newly created **GwxLocation** object on success, NULL if the operation failed. (See also **GwxLocation** and **GwxDynamic**). *Available only in configure mode.*

Function CreateMetafile(filename As String, left As Single, top As Single, width As Single, height As Single, objectName As String) As Object

Creates and returns a **GwxMetafile** object with the specified attributes, in the current display. (See also **GwxVisible**). The **fileName** parameter refers to the file name of a WMF/EMF file. *Available only in configure mode.*

Function CreatePickDynamic(visibleObjectName As String, dynamicObjectName As String, pickAction As GWXPICKACTION, pickType As GWXBUTTONTYPE, executionTrigger As GWXEXECUTIONTRIGGER, mouseButton As GWXMOUSEBUTTON, initiallySelected As Boolean, groupName As String, filename As String, modal As Boolean, center As Boolean, value1 As String, value2 As String, value3 As String) As Object

Creates a **GwxPick** object with the specified attributes and attaches it to the visible object with the specified object name. Valid values for **GWXPICKACTION** are:

PickLoadDisplay	= 0
PickDragDropLoad	= 1
PickPopupWindow	= 2
PickDownloadValue	= 3
PickToggleValue	= 4
PickLaunchApp	= 5
PickClose	= 6
PickRunScript	= 7
PickCustomCommand	= 8
PickEmbeddedWindow	= 9
PickDisplayBack	= 10
PickDisplayForward	= 11
PickSetAliases	= 12
PickAliasesDialog	= 13
PickLayerHideOn	= 14
PickLayerHideOff	= 15
PickLayerHideToggle	= 16
PickPopupUserMenu	= 17
PickSwitchLanguage	= 18
PickSelectThema	= 19
PickLaunchWebPage	= 20

Valid values for **GWXBUTTONTYPE** are:

ButtonNormal	= 0
ButtonCheck	= 1
ButtonRadio	= 2

Valid values for **GWXEXECUTIONTRIGGER** are:

TriggerOnDown	= 1
TriggerWhileDown	= 2
TriggerOnDnWhileDn	= 3
TriggerOnUp	= 4
TriggerOnDnOnUp	= 5
TriggerWhileDnOnUp	= 6
TriggerOnDnWhileDnOnUp	= 7

Valid values for **GWXMOUSEBUTTON** are:

MouseButtonLeft	= 0
MouseButtonMiddle	= 1
MouseButtonRight	= 2

Depending upon the value of **pickAction**, some parameters may be ignored by GraphWorX. Returns the newly created **GwxPick** object on success, NULL if the operation failed. (See also **GwxPick** and **GwxDynamic**). *Available only in configure mode.*

Function CreatePolyline(vertices, isFilled As Boolean, fillColor As OLE_COLOR, lineColor As OLE_COLOR, lineWidth As Long, lineStyle As GWXLINESTYLE, hasShadow As Boolean, shadowColor As OLE_COLOR, edgeStyle As GWX3DEDGESTYLE, isHidden As Boolean, objectName As String) As Object

Creates and returns a **GwxLine** object with the specified attributes, in the current display. The parameter **vertices** is an array of float values such that the elements 0, 2, 4, 6... of the array are x-coordinates of the vertices of the line, and elements 1, 3, 5, 7... of the array are y-coordinates of the vertices of the line. (See also **GwxLine** and **GwxVisible**). When calling this method from a C++ application, the **vertices** parameter should be a "safe array" with the lower bound set to 1. *Available only in configure mode.*

Function CreateProcessPoint(textObjectName As String, dynamicObjectName As String, dataType As GWXDATATYPE, update As Boolean, dataEntry As Boolean, hasInitialValue As Boolean, initialValue, format As String) As Object

Creates a **GwxProcessPoint** object with the specified attributes and attaches it to the **GwxText** object with the specified object name. Returns the newly created **GwxProcessPoint** object on success, NULL if the operation failed. (See also **GwxProcessPoint**, **GwxText**, **GwxVisible**, and **GwxDynamic**). *Available only in configure mode.*

Function CreateRectangle(left As Single, top As Single, width As Single, height As Single, isFilled As Boolean, fillColor As OLE_COLOR, lineColor As OLE_COLOR, lineWidth As Long, lineStyle As GWXLINESTYLE, hasShadow As Boolean, shadowColor As OLE_COLOR, edgeStyle As GWX3DEDGESTYLE, isHidden As Boolean, objectName As String, rounded As Boolean) As Object

Creates and returns a **GwxRectangle** object with the specified attributes, in the current display. (See also **GwxRectangle** and **GwxVisible**). *Available only in configure mode.*

Function CreateRotationDynamic(visibleObjectName As String, dynamicObjectName As String, startAngle As Single, endAngle As Single, pivotX As Single, pivotY As Single, clockwise As Boolean, dial As Boolean, tracking As Boolean, numberOfDetents As Integer, continuousUpdate As Boolean) As Object

Creates a **GwxRotation** object with the specified attributes and attaches it to the visible object with the specified object name. The parameters **pivotX** and **pivotY** are offsets from the center of the object. Angles are specified in degrees. Returns the newly created **GwxRotation** object on success, NULL if the operation failed. (See also **GwxRotation** and **GwxDynamic**). *Available only in configure mode.*

Function CreateSizeDynamic(visibleObjectName As String, dynamicObjectName As String, sizeType As GWXSIZEDYNTYPE, clip As Boolean, startSize As Single, endSize As Single) As Object

Creates a **GwxSize** object with the specified attributes and attaches it to the visible object with the specified object name. The parameters **startSize** and **endSize** are percentage values in the range of 0.0 to 1.0. Valid values for **GWXSIZEDYNTYPE** are:

SizeLeft	= 0
SizeRight	= 1
SizeUp	= 2
SizeDown	= 3
SizeUpLeft	= 4
SizeUpRight	= 5
SizeDownLeft	= 6
SizeDownRight	= 7
SizeLeftRight	= 8
SizeUpDown	= 9
SizeLeftRightBias	= 10
SizeUpDownBias	= 11
SizeAllFour	= 12
SizeLeftRightUp	= 13
SizeLeftRightDown	= 14
SizeUpDownLeft	= 15
SizeUpDownRight	= 16

Returns the newly created **GwxSize** object on success, NULL if the operation failed. (See also **GwxSize** and **GwxDynamic**). Available only in configure mode.

Function CreateStateField(textObjectName As String, dynamicObjectName As String, update As Boolean, dataEntry As Boolean, hasInitialValue As Boolean, initialValue, stateConfiguration As String, defaultState As String) As Object

The **stateConfiguration** parameter has the following format:

value1<TAB>statestring1<CR-LF>value2<TAB>statestring2<CR-LF> ... etc.

The **defaultState** parameter is the string shown when no matching state is found.

For more information, see **CreateProcessPoint()**. Available only in configure mode.

Function CreateSymbol(objectName As String) As Object

Groups all currently selected objects into a **GwxSymbol** object, giving the resulting symbol object the specified object name. Returns the resulting **GwxSymbol** object. (See also **GwxSymbol** and **GwxVisible**). Available only in configure mode.

Function CreateText(x As Single, y As Single, text As String, alignment As GWXTEXTALIGNMENT, stretchText As Boolean, isFilled As Boolean, fillColor As OLE_COLOR, lineColor As OLE_COLOR, lineWidth As Long, lineStyle As GWXLINESTYLE, hasShadow As Boolean, shadowColor As OLE_COLOR, edgeStyle As GWX3DEDGESTYLE, isHidden As Boolean, objectName As String) As Object

Creates and returns a **GwxText** object with the specified attributes, in the current display. (See also **GwxText** and **GwxVisible**). Available only in configure mode.

Function CreateTimedate(textObjectName As String, dynamicObjectName As String, formatType As GWXTIMEDATEFORMATTYPE, timeFormat As String, dateFormat As String) As Object

Creates a **GwxTimedate** object with the specified attributes and attaches it to the **GwxText** object with the specified object name. Returns the newly created **GwxTimedate** object on success, NULL if the operation failed. (See also **GwxTimedate**, **GwxText**, **GwxVisible**, and **GwxDynamic**). Available only in configure mode.

Function DeleteDynamic(objectName As String) As Boolean

Deletes the dynamic object with the specified object name. Returns TRUE for success, FALSE for failure. *Available only in configure mode.*

Function DeleteObject(objectName As String) As Boolean

Deletes the visible object with the specified object name. Returns TRUE for success, FALSE for failure. *Available only in configure mode.*

Sub DeselectAllObjects()

Deselects all currently selected visible objects. *Available only in configure mode.*

Function DisplayBack() As Boolean

Opens the previously loaded display in the display file history. *(Not available in ActiveX ToolWorX).*

Function DisplayForward() As Boolean

Opens the next display in the display file history. *(Not available in ActiveX ToolWorX).*

Sub DuplicateSelection()

Creates duplicates of the currently selected objects. Available only in configure mode. (Not available for displays running in GWXview objects).

Sub EnableRuntimeEvents(LeftButton As Boolean, MiddleButton As Boolean, RightButton As Boolean, MouseMove As Boolean, Keyboard As Boolean)

This method enables or disables firing of keyboard and mouse events into VBA.

LeftButton

Set it to True to get DisplayMouse events related to the left button actions.

MiddleButton

Set it to True to get DisplayMouse events related to the middle button actions.

RightButton

Set it to True to get DisplayMouse events related to the right button actions.

MouseMove

Set it to True to get DisplayMouseMove events.

Keyboard

Set it to True to get Keyboard events.

A typical use of this method:

```
Private Sub GwxDisplay_PreRuntimeStart()
    ' Enable right click events only
    Call ThisDisplay.EnableRuntimeEvents( _
        False, False, True, False, False)
End Sub
```

Sub ExitApplication()

Exits GraphWorX. *(Not available for displays running in GWXview objects).*

Function ExportBitmapCaptureOfView(filename As String) As Boolean

Exports a bitmap (BMP) file with specified name. The bitmap is an image of the current GraphWorX view. *(Not available for displays running in GWXview objects).*

Function FileNew() As Boolean

Starts a new (empty) display. Returns TRUE for success, FALSE for failure.

Function FileOpen(filename As String) As Boolean

Loads the specified display file. Returns TRUE for success, FALSE for failure. (In ActiveX ToolWorX this method only opens resource displays, not disk file displays).

Function FileOpenSetAliases(filename As String, aliasCommandType As GWXALIASCOMMAND, aliasCommandString As String) As Boolean

Opens the specified file and initially sets the specified aliases. If currently in runtime mode, the aliases will be set before any tags are requested. The parameter **aliasCommandType** can be one of the following values:

AliasNoCommand	= 0
AliasSetFromFile	= 1
AliasSetDirect	= 2

If **aliasCommandType** is **AliasSetFromFile**, **aliasCommandString** should be a filename that specifies a text file containing tab-separated pairs of alias names and alias definitions (see also **SetAliasesFromFile()**). If **aliasCommandType** is **AliasSetDirect**, **aliasCommandString** should have the following format:

aliasName1<TAB>aliasDef1<CR-LF>aliasName2<TAB>aliasDef2<CR-LF> ... etc.

(In ActiveX ToolWorX this method only opens resource displays, not disk file displays).

Function FilePrint() As Boolean

Prints the current display. Returns TRUE for success, FALSE for failure. (Not available in ActiveX ToolWorX).

Function FileSave() As Boolean

Saves the current display. Returns TRUE for success, FALSE for failure. (Not available for displays running in GWXview objects).

Function FileSaveAs(filename As String) As Boolean

Saves the current display using the specified file name. Returns TRUE for success, FALSE for failure. (Not available for displays running in GWXview objects).

Function FileSaveAsType(filename As String, displayType As Integer) As Boolean

Saves the current display using the specified file name. The value of displayType should be zero for a VBA-display or one for a non-VBA-display. Returns TRUE for success, FALSE for failure. (Not available for displays running in GWXview objects).

Sub GetClientDimensionsPixels(left As Long, top As Long, width As Long, height As Long)

Gets the GraphWorX client rectangle's size and location in pixels. The client rectangle is the area of the main window not including the borders, title bar, and menu bar.

Sub GetDisplayDimensions(width As Long, height As Long)

Retrieves the display dimensions (work area/world bounds) of the currently loaded display.

Function GetDynamicObjectFromName(objectName As String) As Object

Gets the dynamic object with the specified object name. The returned object is of type **GwxDynamic** or one of the **GwxDynamic** derived types (**GwxSize**, **GwxLocation**, **GwxRotation**, **GwxHide**, **GwxFlash**, **GwxPick**, **GwxDigitalColor**, **GwxAnalogColor**, **GwxAnimator**, **GwxAnalogSelector**, **GwxDigitalSelector**, **GwxProcessPoint**, **GwxTimedate**). Returns NULL if no matching object is found.

Function GetEventScriptText(Event As GWXEVENTNAME) As String

GraphWorX 7.0 allows for associating VBScript and JScript with events. The scripts associated with an event are executed when the event is triggered (for example, after runtime starts, or before animating a layer). These scripts can be edited through the new integrated script toolbar or through the OLE automation interface. Each script can be written using VBScript or JScript language independently from the other event. This function allows for reading (not changing) the script source code currently associated with the specified event.


```

enum {
    DataEntryActivated           = 0,
    DataEntryDeactivated        = 1,
    DataEntryValueEntered       = 2,
    DialBeginDrag               = 3,
    DialDrag                    = 4,
    DialEndDrag                 = 5,
    DisplayLoad                 = 6,
    DisplayUnload               = 7,
    LanguageChanged             = 8,
    PickPostDown               = 9,
    PickPostUp                  = 10,
    PickPostWhileDown          = 11,
    PickPreDown                 = 12,
    PickPreUp                   = 13,
    PickPreWhileDown           = 14,
    PostAnimateDisplay          = 15,
    PostDeanimateDisplay       = 16,
    PreAnimateDisplay           = 17,
    PreDeanimateDisplay        = 18,
    PreAnimateLayer             = 19,
    PostAnimateLayer            = 20,
    PreDeanimateLayer          = 21,
    PostDeanimateLayer         = 22,
    PreRuntimeStart             = 23,
    PostRuntimeStart            = 24,
    PreRuntimeStop              = 25,
    PostRuntimeStop             = 26,
    SliderBeginDrag             = 27,
    SliderEndDrag               = 28,
    SliderDrag                   = 29
} GWXEVENTNAME;

```

Function *GetEventScriptType(Event As GWXEVENTNAME) As GWXSCRIPTTYPE*

GraphWorX 7.0 allows for associating VBScript and JScript with events. The scripts associated with an event are executed when the event is triggered (for example, after runtime starts, or before animating a layer). These scripts can be edited through the new integrated script toolbar or through the OLE automation interface. Each script can be written using VBScript or JScript language independently from the other event. This function allows for reading (not changing) the script source code currently associated with the specified event.

```

enum {
    ScriptTypeNone              = -1,
    ScriptTypeVBA                = 0,
    ScriptTypeVBScript           = 1,
    ScriptTypeJScript            = 2
} GWXSCRIPTTYPE;

```

```

'VBA example, works in configure and runtime mode
'VBA example creates VB script on event PostRuntimeStart

'sets PostRuntimeStart event script type to VB script
ThisDisplay.SetEventScriptType _
    PostRuntimeStart, ScriptTypeVBScript
'adds text to the script

```

```
ThisDisplay.SetEventScriptText PostRuntimeStart, _  
    "MsgBox " & Chr(34) & "Test OK" & Chr(34)  
MsgBox ThisDisplay.GetEventScriptType(PostRuntimeStart)  
MsgBox ThisDisplay.GetEventScriptText(PostRuntimeStart)
```

Function GetFileContainsVBA() As Boolean

The function **GetFileContainsVBA()** returns True value if the display currently contains the VBA part and False if the display does not contain the VBA part. The presence of the VBA part is controlled by loading a non-VBA display and saving the display as non-VBA, or by using the **FileSaveAsType(1)** (contains VBA) or **FileSaveAsType(0)** (does not contain VBA) automation methods.

Function GetFileName() As String

Returns the name of the currently loaded display.

Function GetFilePath() As String

Returns the path where the currently loaded display is located. The returned path does not include the file name of the currently loaded display.

Function GetFileVersion() As Long

The function **GetFileVersion()** returns the following values:

- -1.. OpenFileError (&HFFFFFFF)
- 0 .. Ver520
- 1 .. Ver600
- 2 .. Ver600SP
- 3 .. VerMGraphics40
- 4 .. Ver610
- 9999 .. VerLatest (&H270F)

Function GetFrameWindowHandle() As Long

Gets the window handle of the frame window in which this **GwxDisplay** object is running. (Not available in ActiveX ToolWorX).

Function GetGASThemes() As String

Gets the currently set Global Aliasing themes in the same format as generated by the Theme browser.

```
'VBA/VB Script example, works only in runtime mode  
'example for setting and getting themes (Floors theme is changed in  
child windows)  
ThisDisplay.SetGASThemes _  
    "Buildings=BuildingB;Floors=Floor2/+1;"  
MsgBox ThisDisplay.GetGASThemes
```

Function GetHeadObject() As Object

Gets the head visible object. In configure mode, the head object is the selected object with the solid blue or red resize grips. In runtime mode, the head object is the pickable object which currently has focus. If there is no head object, this method returns NULL.

Function GetLoadTab(index As Long, label As String, File As String, AliasCmdType As Long, AliasCmdString As String, GlobalAliasCmdString As String) As Boolean

Returns TRUE when LoadTab information is successfully retrieved. The index parameter is zero-based index of the load tab. Load Tab information is returned via the set of remaining parameters involved.

Function GetLoadTabsCount() As Long

Returns number of Load Tabs in the display.

Function GetNumberOfTopLevelVisibles() As Long

Returns the number of visible objects in the root symbol of a display (referred to as "Top Level Visibles" because they are the objects in top-most level of the symbol hierarchy).

Function GetTypeNames() As String

Returns the type name of this object ("GwxDisplay").

Function GetOpenPopupByIndex(popupIndex As Long) As Object

Returns the **GwxDisplay** object of the open popup or embedded popup window with the specified zero-based index (**popupIndex**, indexes the list of popups and embedded popups owned by this display). (Not available in ActiveX ToolWorX).

Function GetOpenPopupCount() As Long

Returns the number of open popup and embedded popup windows owned by this display. (Not available in ActiveX ToolWorX).

Function GetOpenPopupWithTitle(titleSubstring As String) As Object

Gets the open popup window's display object which has the specified substring as part or all of the text in the popup window's title bar. Returns the object of the display loaded into the popup window (of type **GwxDisplay**) on success, NULL on failure. (Not available in ActiveX ToolWorX).

Function GetParentDisplay() As Object

If this display object is running in a popup window or embedded window, this method returns the display object of the parent display (the parent display is the display that opened the popup window). If the display is not running in a popup/embedded window, this method returns NULL. (Not available in ActiveX ToolWorX).

Function GetPointObjectFromName(pointName As String) As Object

Gets the data point object with the specified point name. The returned object is of type **GwxPoint**. Returns NULL if no matching object is found.

Function GetSystemWideLanguage(Language As String, LanguageSubset As String, LocaleID As Long, SynchronizeLCID As Boolean) As Boolean

Gets the currently selected language for ProcessView applications.

Function GetTransparency(TransparencyEnabled As Boolean, transparentColor As OLE_COLOR, AlphaEnabled As Boolean, AlphaBlending As Integer) As Boolean

Gets information about the transparency settings. This function works only in GraphWorX on Windows 2000 and higher.

```
'VBA example, works in configure and runtime mode
'this VBA example gets transparency and translucency settings
```

```
Dim TE As Boolean, _
    TC As OLE_COLOR, _
    AE As Boolean, _
    AB As Integer

ThisDisplay.GetTransparency TE, TC, AE, AB
MsgBox "Transparency: " & TE & vbCrLf & "Color: " & _
    TC & vbCrLf & "Translucency: " & AE & _
    vbCrLf & "Color: " & AB
```

void GetViewDimensions(long* left, long* top, long* width, long* height)

Gets the GraphWorX view rectangle's size and location.

Function GetVisibleObjectFromIndex(index As Long) As Object

Returns the Top Level Visible with the given zero-based index. An object with the an index of zero is the object furthest back in the z-order. This function is useful for iterating through all the top level visible objects.

Function GetVisibleObjectFromName(objectName As String) As Object

Gets the visible object with the specified object name. The returned object type will be **GwxVisible** or one of **GwxVisible** derived object types (**GwxRectangle**, **GwxEllipse**, **GwxLine**, **GwxText**, **GwxArc**, **GwxBitmap**, **GwxMetafile**, **GwxSymbol**, **GwxOleObject**, **GwxButton**). Returns NULL if no matching object is found.

Function GetVisibleObjectFromPosition(x As Long, y As Long, GoIntoSymbol As Boolean) As Object

Returns an object at a specified position in the display. Starts from the root of the display and goes deeper, including layers, until the top-most object is found. If the argument **GoIntoSymbol** is set to True, the searching continues until the final visible object at a given position. Otherwise the function returns the top-level symbol that contains the object.

```
'VBA example, works in configure and runtime mode
'this example gets the lowest level visible at random position

Dim w As Long, h As Long
Dim o As Object
ThisDisplay.GetDisplayDimensions w, h
'gets display width and height
If ThisDisplay.GetNumberOfTopLevelVisibles > 0 Then
'checks if display has at least one visible object
! and runs a loop for find one
Do
Set o = ThisDisplay.GetVisibleObjectFromPosition( _
    Rnd * w, Rnd * h, True)
'tries to get visible object from random position - _
!if object is GwxSymbol, it goes into symbol and _
!continues searching
Loop Until Not (o Is Nothing)
'after any object is found, loop ends and example _
!shows name of the object
MsgBox o.objectName
Else
MsgBox "Example needs at least one visible object"
End If
```

Sub GetWindowDimensionsPercent(left As Single, top As Single, width As Single, height As Single)

Gets the GraphWorX main window size and location as a percentage of the total screen size. Retrieved parameters will have values in the range of 0.0 to 1.0.

Sub GetWindowDimensionsPixels(left As Long, top As Long, width As Long, height As Long)

Gets the GraphWorX main window size and location in pixels.

Function GetWorkingDirectory() As String

Returns the ProcessView working directory. This is the directory where ProcessView applications like GraphWorX look for display files and other configuration files.

Function GwxGetVbaProject() As Object

Returns the VBA Project interface. (Not available for displays running in GWXview objects).

Sub HideWindow()

Hides the GraphWorX main window.

Function InsertLibraryObject(libraryObject As Object) As Boolean**Function InsertLibraryObject(libraryObject As Object) As Boolean**

Inserts a Symbol Library object into display. This library object can be created/obtained via Symbol Library OLE automation. (See the Symbol Library Help documentation for more information.)

Note: This function works in GraphWorX design mode only.

Function InsertLibraryObjectByIndex(category As String, index As Long) As Boolean

Allows easy inserting of any symbol from the Symbol Library via OLE automation. The **category** parameter corresponds with the name of the Symbol Library category file (.sdf file); the parameter **index** is zero-based index of the symbol in the category.

Note: This function works in GraphWorX design mode only.

Function InsertLibraryObjectByName(category As String, symbol As String) As Boolean

Allows easy inserting of any symbol from the Symbol Library via OLE automation. The **category** parameter corresponds to the name of the Symbol Library category file (.sdf file); the parameter **symbol** is the name of the symbol from category file.

Note: This function works in GraphWorX design mode only.

Typical use of this method:

```
Dim SL As SymbolLibrary.SymbolLibrary
Dim SC As SymbolLibrary.SymbolCategory
Dim o As Object
Set SL = New SymbolLibrary.SymbolLibrary
Set SC = SL.GetCategoryFromIndex(0)
If SC.GetNumberOfSymbols > 0 Then
  ThisDisplay.InsertLibraryObjectByIndex _
    Dir(SC.GetCategoryName), _
    Round(Rnd * SC.GetNumberOfSymbols)
  ThisDisplay.InsertLibraryObjectByName _
    Dir(SC.GetCategoryName), _
    SC.GetSymbolName(Round(Rnd * SC.GetNumberOfSymbols))
  Set o = SC.GetSymbolFromIndex( _
    Round(Rnd * SC.GetNumberOfSymbols))
  ThisDisplay.InsertLibraryObject o
Else
  MsgBox "Example needs at least one symbol"
End If
```

Function InsertOleObject(progID As String, objName As String) As Boolean

This function allows you to create an OLE Automation object and then insert that object into your GraphWorX displays. This automation interface can be called through VBA or VBScript/JScript or C++. This method requires that you provide the name of the object that you wish to create. You can create any kind of object that you wish but you have to know the name. For example, SMAR uses the following objects:

- "SMAR.SCRAAlarmCtrl.1"
- "SMAR.Event32.1"
- "SMAR.Periodic32.1"
- "GWXGAUGE.GWXGaugeCtrl.1"
- "GWXSWITCH.GWXSwitchCtrl.1"
- "TWXViewer.TWXViewerCtrl.1"
- "AWXview32.AWXview32Ctrl.1"
- "GWXVIEW32.GWXview32Ctrl.1"
- "GWXNUMERIC.GWXNumericCtrl.1"
- "GWXVESSEL.GwxVesselCtrl.1"
- "SMAR.FunctionalKeyCtrl.1"
- "MarqueeLaunch.MarqueeCtl.1"
- "PopupLaunch.PopupCtl.1"

Function IsEmbeddedPopupWindow() As Boolean

TRUE if this display is running in an embedded popup window; FALSE otherwise. (Not available in ActiveX ToolWorX).

Function IsModified() As Boolean

Returns TRUE is the display has been modified since the last time it was saved.

Function IsPopupWindow() As Boolean

TRUE if this display is running in a popup window or an embedded popup window; FALSE otherwise. (Not available in ActiveX ToolWorX).

Function IsRuntimeMode() As Boolean

Returns TRUE is GraphWorX is currently in runtime mode, FALSE otherwise.

Sub LogicalPointToClient(x As Long, y As Long)

Converts GraphWorX logical coordinates to client coordinates. *(Not available in ActiveX ToolWorX).*

Sub LogicalPointToScreen(x As Long, y As Long)

Converts GraphWorX logical coordinates to screen coordinates. *(Not available in ActiveX ToolWorX).*

Sub MaximizeWindow()

Maximizes the GraphWorX main window. *(Not available for displays running in GWXview objects).*

Sub MinimizeWindow()

Minimizes the GraphWorX main window. *(Not available for displays running in GWXview objects).*

Function MoveSelectionBackward() As Boolean

Moves the selected object backward by one position. It is used to visually sort objects and to put one object below another.

Function MoveSelectionForward() As Boolean

Moves the selected object forward by one position. It is used to visually sort objects and to put one object above another.

Function MoveSelectionToBack() As Boolean

Move the selected object behind all of the other objects in the display.

Function MoveSelectionToFront() As Boolean

Move the selected object in front of all of the other objects in the display.

Function OpenEmbeddedWindow(filename As String, center As Boolean, hidden As Boolean) As Object

Opens a GraphWorX embedded window. Returns the object of the display loaded into the embedded window (of type **GwxDisplay**) on success, NULL on failure. *(Not available in ActiveX ToolWorX).*

Function OpenEmbeddedWinSetAliases(filename As String, center As Boolean, hidden As Boolean, aliasCommandType As GWXALIASCOMMAND, aliasCommandString As String) As Object

Opens an embedded popup window with the specified display file and initially sets the specified aliases. If currently in runtime mode, the aliases will be set before any tags are requested. For descriptions of **aliasCommandType** and **aliasCommandString**, see **FileOpenSetAliases()**. *(Not available in ActiveX ToolWorX).*

Function OpenPopupWindow(filename As String, modal As Boolean, center As Boolean, hidden As Boolean) As Object

Opens a GraphWorX popup window. Returns the object of the display loaded into the popup window (of type **GwxDisplay**) on success, NULL on failure. *(Not available in ActiveX ToolWorX).*

Function OpenPopupWinSetAliases(filename As String, modal As Boolean, center As Boolean, hidden As Boolean, aliasCommandType As GWXALIASCOMMAND, aliasCommandString As String) As Object

Opens a popup window with the specified display file and initially sets the specified aliases. If currently in runtime mode, the aliases will be set before any tags are requested. For descriptions of **aliasCommandType** and **aliasCommandString**, see **FileOpenSetAliases()**. *(Not available in ActiveX ToolWorX).*

Function OpenSetAliasesDialog(aliasCommandType As GWXALIASCOMMAND, aliasCommandString As String, objectName As String) As Boolean

Opens the Set Aliases Dialog. This dialog is used to allow the operator to enter new alias definitions for a set of specified alias names. For descriptions of **aliasCommandType** and **aliasCommandString**, see **FileOpenSetAliases()**. The **aliasCommandString** parameter is the primary means of configuring the dialog. For each unique alias name specified in **aliasCommandString**, there will be a tab added to the dialog box. Alias definitions specified with

the alias names will be listed in the appropriate property page's combo box. The **objectName** parameter specifies which visible object the new alias settings will apply to (if no object name is specified, new alias settings will be applied to the entire display). (Not available in ActiveX ToolWorX).

Function OpenTagBrowser(hWndParent As Long, tagName As String) As Boolean

Opens the Tag Browser with the specified window handle as the parent window. The parameter **tagName** receives the tag name selected by the user. Returns TRUE if the user hit the OK button, FALSE if the user hit the Cancel button.

Function PopAllCurrentSymbol() As Boolean

Pops the current symbol edit level all the way back to the root level. Returns TRUE for success, FALSE for failure. (See also **GwxSymbol::PushCurrentSymbol()**). Available only in configure mode.

Function PopCurrentSymbol() As Boolean

Pops the current symbol edit level up one level. Returns TRUE for success, FALSE for failure. (See also **GwxSymbol::PushCurrentSymbol()**). Available only in configure mode.

Function QueryDataType(dataSource As String, dataType As GWXDATATYPE) As Boolean

Queries the OPC server for the data type of the specified tag name (**dataSource**). See **GwxPoint** for possible values of GWXDATATYPE.

Function QueryRanges(dataSource As String, lowRange As Double, highRange As Double) As Boolean

Queries the OPC server for the high and low range values of the specified tag name (**dataSource**).

Sub RefreshWindow()

Redraws the visible portion of a GraphWorX display.

Sub RemoveLayer(layerName As String)

Removes the specified layer from the display.

Function ReplaceFilePath(OldSubstring As String, NewSubstring As String) As Long

Works over path name parameters of pick dynamics. Returns 0 on success and HRESULT when something fails.

Function ReplaceHost(OldHostName As String, NewHostName As String) As Long

Works over both data source tags and path name attributes and replaces node name **substring** within URL path only. Returns 0 on success and HRESULT when something fails.

Function ReplaceHostEx(OldHostNameSubstring As String, NewHostNameSubstring As String, MatchCase As Boolean, MatchWholeWord As Boolean) As Long

Works over both data source tags and path name attributes and replaces node name **substring** within URL path only, and supports case-sensitivity, wildcard strings and MatchWholeWord flag. Returns 0 on success and HRESULT when something fails.

Function ReplaceTag(OldSubstring As String, NewSubstring As String) As Long

Works over data source tags in dynamic actions of all display objects. Replaces **OldSubstring** with **NewSubstring** in certain situations. Returns 0 on success and HRESULT when something fails.

Typical use of this method:

```
'VBA example, works only in configure mode
'replaces tag name in whole display

Dim Status As Long
Status = ThisDisplay.ReplaceTag("gfwsim.ramp.float",
"gfwsim.sine.double")
'if Status <> 0 then there was no replacement
'performed or an error occurs
If Status <> 0 Then
    MsgBox "No replacements"
Else
```

```
MsgBox "Tags replaced"  
End If
```

long ReplaceStringInString (BSTR StringToReplace, BSTR OldSubstring, BSTR NewSubstring)

Replaces **OldSubstring** with **NewSubstring** inside (within) **StringToReplace**. Returns 0 on success and HRESULT when something fails.

Sub RestoreWindow()

Restores the (non-minimized/non-maximized) window size and position. *(Not available for displays running in GWXview objects).*

Function SelectPaletteColor(SelectedColor As OLE_COLOR) As Boolean

Opens the GraphWorX color palette dialog box. The parameter **SelectedColor** receives the color selected by the user. Returns TRUE if the user hit the OK button, FALSE if the user hit the Cancel button. *(Not available for displays running in GWXview objects).*

Function SelectPaletteColorRGB(red, green, blue) As Boolean

Pops up a dialog with a color palette and allows you to select one color between them.

Returns TRUE if a color is selected; returns FALSE in all other cases.

Each color displayed in the computer monitor can be decomposed into 3 components: red, green and blue. Each of these component colors can have an intensity that ranges from 0 (the component is not present) to 255 (the maximum that each component color can contribute to a color).

If the user selects a color, the 3 parameters are filled with the red, green and blue composition of the selected color. For example, if you select a bright yellow color, then you will get back the following:

```
Red=255  
Green=255  
Blue=0
```

This method works in Visual Basic for Applications only if the parameters are declared as Variant.

Function SetAliasDefinition(aliasName As String, newDefinition As String) As Boolean

Sets the alias definition of the specified alias name for all dynamic objects in the display. This function can be used in runtime mode to easily change the data connections of dynamic objects on-the-fly. Returns FALSE if no matching alias names were found, TRUE otherwise. (See also **GwxVisible::SetAliasDefinition** and **GwxDynamic::SetAliasDefinition**).

Function SetAliases(commandString As String) As Boolean

Sets multiple alias definitions for the entire display. The **commandString** parameter has the following format:

```
aliasName1<TAB>aliasDef1<CR-LF>aliasName2<TAB>aliasDef2<CR-LF> ... etc.
```

Function SetAliasesFromFile(filename As String) As Boolean

Sets multiple alias definitions for the entire display. The **filename** parameter specifies a text file containing tab-separated pairs of alias names and alias definitions (the format of this file is the same format as the string passed to **GwxDisplay's SetAliases()** method). Files of this type can be created in **Notepad**, can be exported from **Excel** (save as Tab Delimited), and can be created using **Gwx's Alias File Editor**.

Sub SetDisplayDimensions(width As Long, height As Long)

Sets the display dimensions (work area/world bounds) of the currently loaded display.

Sub SetEventScriptText(Event As GWXEVENTNAME, text As String)

GraphWorX 7.0 allows for associating VBScript and JScript with events. The scripts associated with an event are executed when the event is triggered (for example, after runtime starts, or before animating a layer). These scripts can be edited through the new integrated script toolbar or through the OLE automation interface. Each script can be written using VBScript or JScript language independently from the other event. This function allows for writing (changing) the script source code currently selected for the specified event.

Sub SetEventScriptType(Event As GWXEVENTNAME, Type As GWXSCRIPTTYPE)

GraphWorX 7.0 allows for associating VBScript and JScript with events. The scripts associated with an event are executed when the event is triggered (for example, after runtime starts, or before animating a layer). These scripts can be edited through the new integrated script toolbar or through the OLE automation interface. Each script can be written using VBScript or JScript language independently from the other event. This function allows for writing (changing) the script source code currently associated with the specified event.

```
'VBA example, works in configure and runtime mode
'VBA example creates VB script on event PostRuntimeStart

ThisDisplay.SetEventScriptType _
    PostRuntimeStart, ScriptTypeVBScript
ThisDisplay.SetEventScriptText _
    PostRuntimeStart, "MsgBox " & Chr(34) & _
    "Test OK" & Chr(34)
```

Corresponding VBScript Example

```
ThisDisplay.SetEventScriptText 23,"msgbox ""test"" "
ThisDisplay.SetEventScriptType 23,1
```

Function SetFileVersion(NewFileVersion As GWXSETFILEVERSION) As GWXGETFILEVERSION

The function **SetFileVersion()** sets a display version for the next save. Note that the set of versions supported for saving is less than the set of versions detected on reading the file. In other words, GraphWorX detects more versions of files, but it allows saving only in the more recent versions; version 5.20 and older versions are not supported. If there is an attempt to save in an unsupported older version, then GraphWorX saves in the version SetVer600.

SetFileVersion() input values

- 2 .. SetVer600SP
- 3 .. SetVerMGraphics40
- 4 .. SetVer610
- 9999 .. SetVerLatest (&H270F)

The automation interface is very powerful, and it can be used for automated display processing (e.g. there is a need to upgrade or downgrade a group of displays to a certain version, or to make certain changes in the displays while preserving the existing version).

Visual Basic Code Sample

```
Private Sub DisplayVersion()
    ' Specify the path and filename of an existing display
    Const strPath As String = _
        "C:\PathToYourDisplay\"
    Const strFilename As String = _
        strPath & "YourDisplay.gdf"
    Const SetVer600SP as Long = 2
    Const VerLatest as Long = 9999

    ' Create GWX instance
    Dim g As New Gwx32.GwxDisplay

    ' Check if GWX was created
    If g Is Nothing Then
        MsgBox "Failed to create GWX"
        Exit Sub
    End If

    ' Show GWX in the front and load the requested display
```

```
Call g.ShowWindow
Call g.BringWindowToTop
Call g.FileNew
Call g.FileOpen(strFilename)

' Check the file version
Dim ver As long
ver = GetFileVersion(g)

' If this is the latest version, save in the
' version 6.00
If (ver = VerLatest) Then
    g.SetFileVersion(SetVer600SP)
    g.FileSave
End If

' Exit GWX, the FileNew suppresses the message box
' if to save changes.
g.FileNew
g.ExitApplication
End Sub

' Get file version and trace the current version in the
' Immediate window for debugging purposes.
Private Function GetFileVersion(g As GwxDisplay) As Long
    Const OpenFileError as Long = -1
    Const Ver520 as Long = 0
    Const Ver600 as Long = 2
    Const Ver600SP as Long = 2
    Const VerMGraphics40 as Long = 2
    Const Ver610 as Long = 2
    Const VerLatest as Long = 9999

    Dim ver As long

    ' Check the version
    ver = g.GetFileVersion
    Select Case (ver)
        Case OpenFileError:
            Debug.Print "Version: FileOpenError"
        Case Ver520:
            Debug.Print "Version: Ver520"
        Case Ver600:
            Debug.Print "Version: Ver600"
        Case Ver600SP:
            Debug.Print "Version: Ver600SP"
        Case VerMGraphics40:
            Debug.Print "Version: VerMGraphics40"
        Case Ver610:
            Debug.Print "Version: Ver610"
        Case VerLatest:
            Debug.Print "Version: VerLatest"
    End Select
```

```

' Check if the display contains VBA
If g.GetFileContainsVBA Then
    Debug.Print "VBA: Yes"
Else
    Debug.Print "VBA: No"
End If

GetFileVersion = ver
End Function

```

Function SetGASThemes(Themes As String) As Boolean

Sets Global Alias themes. The format is the same as generated by the Theme Browser, so it is recommended to use the Theme Browser to generate the desired combination and copy/paste the result into this function as string.

```

'VBA/VB Script example, works only in runtime mode
'example for setting and getting themes (Floors theme is
changed in child windows)
ThisDisplay.SetGASThemes
"Buildings=BuildingB;Floors=Floor2/+1;"
MsgBox ThisDisplay.GetGASThemes

```

Function SetGradientFill(style As GWXGRADIENTSTYLE, color1 As OLE_COLOR, color2 As OLE_COLOR, isTwoColor As Boolean, brightness As Single, reverseColors As Boolean, steps As Integer, horizontalOffset As Single, verticalOffset As Single) As Boolean

Sets a gradient fill for display background.

The parameter **style** can be one of the following values:

```

GradientHorizontal    = 0
GradientVertical      = 1
GradientSquare        = 2

```

The rest of the parameters correspond to those in the gradient fill configuration user interface.

Function SetHeadObject(objectName As String) As Boolean

Sets the head object to the visible object with the specified name. Returns TRUE for success, FALSE for failure.

Function SetLayerOverrideHide(layerName As String, hidden As Boolean) As Boolean

Sets the layer override hide property of the specified display layer.

Sub SetModifiedFlag(Modified As Boolean)

The modified flag is used to determine if the display has been modified since the last time it was saved. If the modified flag is set to true, then GraphWorX will request to save the changes to the display before closing the display or loading a new display.

Function SetRuntimeFocusProperties(ShowHandCursor As Boolean, ShowFocusRectangle As Boolean, fillColor As OLE_COLOR, BorderColor As OLE_COLOR) As Boolean

Sets the attributes for runtime focus highlights.

Function SetRuntimeUpdateRate(updateRate As Long) As Boolean

Sets the OPC data update rate for the display.

Function SetSystemWideLanguage(Language As String, LanguageSubset As String, LocaleID As Long, SynchronizeLCID As Boolean) As Boolean

Sets the ProcessView language and language subset for the document, the LocaleID for application, and the flag if to synchronize document language with an application.

Function SetTooltipProperties(ShowDynamicTips As Boolean, ShowDynamicObjectName As Boolean, ShowDynamicObjectDescription As Boolean, ShowDataSourceName As Boolean,

ShowDataSourceValue As Boolean, ShowStaticTooltips As Boolean, ShowStaticObjectName As Boolean, ShowStaticObjectDescription As Boolean) As Boolean

Sets the tooltip properties for the display.

Function SetTransparency(EnableTransparency As Boolean, transparentColor As OLE_COLOR, EnableAlpha As Boolean, AlphaBlending As Integer) As Boolean

Sets display transparency in runtime. Works only in GraphWorX on Windows 2000 and higher.

```
'VBA example, works in runtime mode
'this example toggles transparency and translucency settings

Dim TE As Boolean, TC As OLE_COLOR, AE As Boolean, _
    AB As Integer
ThisDisplay.GetTransparency TE, TC, AE, AB
'following lines enable/disable transparency and translucency
If TE Then TE = False Else TE = True
If AE Then AE = False Else AE = True
'if translucency is enabled, sets alpha blending value to 100 of 255
If AE Then AB = 100 Else AB = 0
'applies changed values
ThisDisplay.SetTransparency TE, TC, AE, AB
MsgBox "Transparency: " & TE & vbCrLf & "Color: " & _
    & TC & vbCrLf & "Translucency: " & _
    AE & vbCrLf & "Color: " & AB
```

Sub SetViewDimensions(left As Long, top As Long, width As Long, height As Long)

Sets the GraphWorX view rectangle's size and location. The view dimensions define what portion of the work area/whole display is visible. Setting the view dimensions can be used to zoom and pan the view of a display.

Sub SetWindowDimensionsPercent(left As Single, top As Single, width As Single, height As Single)

Sets the GraphWorX main window size and location as a percentage of the total screen size. Parameters should be values in the range of 0.0 to 1.0. (Not available for displays running in GWXview objects).

Sub SetWindowDimensionsPixels(left As Long, top As Long, width As Long, height As Long)

Sets the GraphWorX main window size and location in pixels. (Not available for displays running in GWXview objects).

Sub ShowWindow()

Shows the GraphWorX main window.

Sub StartRuntime()

Puts GraphWorX into runtime mode.

Sub StopRuntime()

Takes GraphWorX out of runtime mode (into configure mode).

Function TestCustomSecurityItem(customString As String) As Boolean

Returns TRUE if the current user has security privilege for the specified custom security item, FALSE otherwise.

Function ToggleLayerOverrideHide(layerName As String) As Boolean

Toggles the layer override hide property of the specified display layer.

Function ToggleRuntime() As Boolean

Toggles the current operator mode between configure mode and runtime mode. Returns TRUE, if the method put GraphWorX into runtime mode, FALSE if the method put GraphWorX into configure mode.

Function UngroupSymbol(objectName As String) As Boolean

Ungroups the **GwxSymbol** object with the specified object name. Although the function destroys the specified **GwxSymbol**, it does not destroy the objects that were grouped in that symbol. Returns TRUE for success, FALSE for failure. *Available only in configure mode.*

Sub ViewBoxZoom()

Enters "Box Zoom" mode. This mode allows the user to stretch a box defining the new view area. *(Not available in ActiveX ToolWorX).*

Sub ViewFitToWindow()

Zooms such that all the objects in the display fit in the view. *(Not available in ActiveX ToolWorX).*

Sub ViewHome()

Zooms to the home view. The home view is that view at which the display was last saved. *(Not available in ActiveX ToolWorX).*

Sub ViewPan(offsetX As Long, offsetY As Long)

Pans (scrolls) the view by the specified offsets. *(Not available in ActiveX ToolWorX).*

Sub ViewShowWholeDisplay()

Zooms such that the entire display dimensions fit in the view. *(Not available in ActiveX ToolWorX).*

Sub ViewUnzoom()

Undoes the previous zoom command. *(Not available in ActiveX ToolWorX).*

Sub ViewZoomCustomDialog()

Brings up the Custom Zoom dialog. This dialog allows the user to type in a new zoom percentage. *(Not available in ActiveX ToolWorX).*

Sub ViewZoomPercent(newZoom As Single)

Zooms in by the specified percentage. *(Not available in ActiveX ToolWorX).*

GwxVisible

GwxVisible is the object type from which visible GraphWorX objects (ellipses, rectangles, etc.) are derived. In other words, all visible GraphWorX objects have the properties and methods of **GwxVisible**.

Properties

Angle As Single

Gets/Sets the rotation angle of the object. The angle is specified in degrees.

edgeStyle As GWX3DEDGESTYLE

The 3-D edge style of the object's border. Valid values for **GWX3DEDGESTYLE** are:

EdgeNone	= 0
EdgeRaised	= 5
EdgeEtched	= 6
EdgeBump	= 9
EdgeSunken	= 10

fillColor As OLE_COLOR

Gets/Sets the visible object's fill color.

hasShadow As Boolean

TRUE if the object has a shadow, FALSE if the object does not have a shadow.

isFilled As Boolean

TRUE if the object is filled, FALSE if the object is not filled.

Keyword As String

This string is used to store a keyword. Use this property to identify an object for GraphWorX's Update Shared Objects functionality.

lineColor As OLE_COLOR

Gets/Sets the visible object's line/border color.

lineStyle As GWXLINestyle

The style of the object's line/border. Valid values for **GWXLINestyle** are:

LineSolid = 0
LineDash = 1
LineDot = 2
LineDashDot = 3
LineDashDotDot = 4
LineNone = 5

lineWidth As Long

The width of an object's line/border. This value must be in the range of 0 to 10.

objectName As String

Gets/Sets the object name of a visible object. The object name is used to identify the object when using certain OLE Automation methods (for instance **GwxDisplay::GetVisibleObjectFromName**). GraphWorX will insure that object names are unique. If you assign an object name that already exists for another visible object in a display, GraphWorX will append a number to the object name (for example, "tank" would become "tank1", "tank1" would become "tank2", etc.).

Selected As Boolean

TRUE if the object is selected, FALSE if the object is not selected. Selected objects are objects which have resize grips around them. *This property can only be changed in configure mode.*

shadowColor As OLE_COLOR

Gets/Sets the visible object's shadow color.

UserCustomData As String

This string is used to store custom data. Use this property to associate any additional data with the visible object.

UserDescription As String

A description string for the visible object. Typically, this string is used to be displayed as informational text in a tooltip.

Visible As Boolean

TRUE if the object is visible, FALSE if the object is hidden.

Methods

Sub ClearGradientFill()

Clears a previously set gradient fill for this visible object.

FlipHorizontal()

Allows for flipping an object on the horizontal axis. The effect is the same as looking at a horizontal mirror.

FlipVertical()

Allows for flipping an object on the vertical axis. The effect is the same as looking at a vertical mirror.

Example Code

```
Dim obj as GwxRectangle
set obj=ThisDisplay.GetVisibleObjectFromName("myobject")
obj.FlipHorizontal
obj.FlipVertical
```

Function GetDataSources(includeDuplicates As Boolean)

Gets all tag names associated with a visible object. Returns a (variant) array of strings (each string in the array is a tag name). The array can optionally include or not include duplicate tag names.

Function GetDynamicObjectFromIndex(index As Long) As Object

Gets the dynamic object with the specified zero-based index, which is attached to this visible object (index zero is the first attached dynamic). This function is useful for iterating through all the dynamic objects attached to this visible object.

Function GetDynamicObjectFromName(nameSubstring As String) As Object

Gets the dynamic object with the specified name (or portion of a name), which is attached to this visible object.

Function GetNumberOfDynamics() As Long

Returns the number of dynamic objects attached to this visible object.

Sub GetObjectDimensions(left As Single, top As Single, width As Single, height As Single)

Gets the object's size and location. This method does not work in VBScript and Jscript.

Sub GetObjectDimensions2(left, top, width, height)

This method works in Visual Basic for Applications as well as VBScript and JScript. To make it work in Visual Basic for Applications, you need to declare the parameters as Variant.

```
'VB example, works only in runtime mode
'example for getting object dimensions (GetObjectDimensions2 returns
variants)
'visible is GwxVisible object
'example requires an object named "test"

Dim l, t, w, h, visible
Set visible = ThisDisplay.GetVisibleObjectFromName("test")
visible.GetObjectDimensions2 l, t, w, h
MsgBox "Left: " & l & vbCrLf & "Top: " & t & _
      vbCrLf & "Width: " & w & _
      vbCrLf & "Height: " & h
```

Function GetObjectTypeName() As String

Returns the type name of this object.

Function GetParentObject() As Object

This method allows you to get a reference point to the parent object in the GraphWorX objects hierarchy. In the following example, imagine you have two objects named "child1" and "child2" grouped together in a symbol named "parent." Starting from the pointer to the child1, you will get a pointer to the parent, and then display its name.

```
Dim obj As GwxRectangle
Set obj = ThisDisplay.GetVisibleObjectFromName("child1")
If obj Is Nothing Then
  MsgBox "can't find child1"
Else
  Set parent = obj.GetParentObject
  If parent Is Nothing Then
    MsgBox "can't get parent"
  Else
    MsgBox parent.objectName
  End If
End If
```

Sub MoveObject(offsetX As Single, offsetY As Single)

Moves the object by the specified offset. *This method only works for objects that are top level visible objects.*

Sub RefreshObject()

Caused the visible object to be redrawn.

Sub Rotate90(clockwise As Boolean)

This method rotates objects by 90 degree. It takes one boolean parameter as input to establish if the rotation is clockwise or not. Follow the simple VBA example below on how to rotate a rectangle by 90 degree clockwise.

```
Dim rect As GwxRectangle
Set rect = ThisDisplay.GetVisibleObjectFromName("rectangle")
If rect Is Nothing Then
    MsgBox "An error has occurred while trying to get the object"
Else
    rect.Rotate90 True
End If
```

Function SetAliasDefinition(aliasName As String, newDefinition As String) As Boolean

Sets the alias definition for all dynamic objects associated with this visible object. If this **GwxVisible** is also a **GwxSymbol**, this function is recursively applied to all objects grouped within the symbol. Returns FALSE if no matching alias names were found, TRUE otherwise. (See also **GwxDisplay::SetAliasDefinition** and **GwxDynamic::SetAliasDefinition**).

Function SetAliases(commandString As String) As Boolean

Sets multiple alias definitions for this visible object. If this **GwxVisible** is also a **GwxSymbol**, this function is recursively applied to all objects grouped within the symbol. (See also, **GwxDisplay::SetAliases()** and **GwxVisible::SetAliasDefinition()**).

Function SetAliasesFromFile(filename As String) As Boolean

Sets multiple alias definitions for this visible object. If this **GwxVisible** is also a **GwxSymbol**, this function is recursively applied to all objects grouped within the symbol. (See also, **GwxDisplay::SetAliasesFromFile()** and **GwxVisible::SetAliasDefinition()**).

Function SetGradientFill(style As GWXGRADIENTSTYLE, color1 As OLE_COLOR, color2 As OLE_COLOR, isTwoColor As Boolean, brightness As Single, reverseColors As Boolean, steps As Integer, horizontalOffset As Single, verticalOffset As Single) As Boolean

Sets a gradient fill for this visible object.

The parameter **style** can be one of the following values:

GradientHorizontal	= 0
GradientVertical	= 1
GradientSquare	= 2

The rest of the parameters correspond to those in the gradient fill configuration user interface.

Sub SetObjectDimensions(left As Single, top As Single, width As Single, height As Single)

Sets the object's size and location. *This method only works for objects that are top level visible objects.*

Sub StretchObject(scaleX As Single, scaleY As Single, anchorX As Single, anchorY As Single)

Stretches the object based on the specified scale factors. The parameters scaleX and scaleY are values such that 0.0 is 0% and 1.0 is 100%. The anchor values are used to change the way the object shifts position during stretching. For example, to resize an object from its center, the anchor values should be the center coordinates of the object. *This method only works for objects that are top level visible objects.*

GwxText

GwxText has all the properties and methods of **GwxVisible**, plus the additional properties described below.

Properties

Font As IFontDisp

Gets/Sets font of the text object.

alignment As GWXTEXTALIGNMENT

The alignment of the text (left, center, right). This property is only significant for multiline text strings. Valid values for **GWXTEXTALIGNMENT** are:

TextAlignLeft	= 0
TextAlignCenter	= 1

TextAlignRight = 2

BoundaryAlignment As GWXTEXTALIGNMENT

The alignment of the text relative to the text object's bounding rectangle (left, center, right). Valid values for **GWXTEXTALIGNMENT** are:

TextAlignLeft = 0

TextAlignCenter = 1

TextAlignRight = 2

stretchText As Boolean

TRUE if the font size should be resized when the text object is stretched. FALSE if the font size should remain the same when the object is stretched.

text As String

This string is the text that is displayed by the text object.

GwxRectangle

GwxRectangle has all the properties and methods of **GwxVisible**, plus the additional properties described below.

Properties

rounded As Boolean

When getting this property, if the value is TRUE, the rectangle has rounded corners. When setting this property to TRUE, the rectangle is given rounded corners with the default rounding settings.

RoundingX As Long

Gets/sets the amount of horizontal rounding of the rectangle's corners.

RoundingY As Long

Gets/sets the amount of vertical rounding of the rectangle's corners.

GwxArc

GwxArc has all the properties and methods of **GwxVisible**, plus the additional properties described below.

Properties

endAngle As Single

Gets/sets the end angle of the arc (in degrees).

arcType As GWXARCTYPE

Gets/sets the arc type of the arc. Valid values for **GWXARCTYPE** are:

ArcArc = 0

ArcPie = 1

ArcChord = 2

startAngle As Single

Gets/sets the start angle of the arc (in degrees).

GwxSymbol

The **GwxSymbol** is used to group together a collection of **GwxVisible** derived objects. **GwxSymbol** objects can be used to group other **GwxSymbol** objects, allowing nested groupings. **GwxSymbol** has all the properties and methods of **GwxVisible**, plus the additional methods described below.

Methods

Function GetDataSourcesRecursive(includeDuplicates As Boolean)

Gets all tag names associated with a symbol (includes all sub-objects). Returns a (variant) array of strings (each string in the array is a tag name). The array can optionally include or not include duplicate tag names.

Function GetNumberOfChildVisibles() As Long

Returns the number of immediate child objects grouped in this symbol.

Function GetVisibleObjectFromIndex(index As Long) As Object

Gets the visible object with the specified zero-based index, which is an immediate child of this symbol (index zero is the first child object). This function is useful for iterating through all the immediate child objects grouped in this symbol.

Function GetVisibleObjectFromName(nameSubstring As String) As Object

Gets the visible object with the specified name (or portion of a name), which is an immediate child of this symbol.

GetVisibleObjectFromPosition(x as long, y as long) as Object

Returns an object at a specified position in the display. Starts from the root of the display and goes deeper including layers, until the top-most object is found. If the argument **GoIntoSymbol** is set to True, the searching continues until the final visible object at a given position. Otherwise the function returns the top-level symbol that contains the object. The symbol function returns the visible object inside the symbol at the specified position.

```
'VBA example, works in configure and runtime mode
'this example gets the lowest level visible at random position

Dim w As Long, h As Long
Dim o As Object
ThisDisplay.GetDisplayDimensions w, h
'gets display width and height
If ThisDisplay.GetNumberOfTopLevelVisibles > 0 Then
'checks if display has at least one visible object
'and runs a loop for find one
Do
Set o = ThisDisplay.GetVisibleObjectFromPosition(Rnd * w, Rnd * h,
True)
'tries to get visible object from random position
'- if object is GwxSymbol, it goes into symbol
'and continues searching
Loop Until Not (o Is Nothing)
'after any object is found, loop ends and example
'shows name of the object
MsgBox o.objectName
Else
MsgBox "Example needs at least one visible object"
End If
```

Function PushCurrentSymbol() As Boolean

Pushes this symbol object onto the symbol edit stack. This essentially makes the objects grouped in this symbol, top level visible objects (some operations, like **GwxVisible::SetObjectDimensions** are only allowed on top level objects).

GwxOleObject

GwxOleObject has all the properties and methods of **GwxVisible**, plus the additional methods described below. (*GwxOleObjects used with ActiveX ToolWorX cannot be activated and consequently should not be used*).

Methods

Function GetOLEObject() As Object

Gets the actual OLE object wrapped by **GwxOleObject**. The method allows access to the embedded ActiveX object. For example, assume that a TrendWorX Viewer object called "twx_01" (the name was assigned in the GraphWorX Property Inspector) is inserted into the GraphWorX display. Then it is possible to use the following code to access this ActiveX control in runtime and work with it:

```
Dim o As GwxOleObject
Set o = ThisDisplay.GetVisibleObjectFromName("twx_01")

Dim twx As TWXVIEWERLib.TWXViewer
Set twx = o.GetOLEObject()

twx.BackColor = RGB(0, 255, 0)
```

GwxEllipse

GwxEllipse has all the properties and methods of **GwxVisible**.

GwxLine

GwxLine has all the properties and methods of **GwxVisible**.

GwxBitmap

GwxBitmap has all the properties and methods of **GwxVisible**.

GwxMetafile

GwxMetafile has all the properties and methods of **GwxVisible**.

GwxButton

A **GwxButton** object is not fully functional until it has at least one **GwxPick** dynamic object associated with it. **GwxButton** has all the properties and methods of **GwxVisible** and **GwxText**.

GwxDynamic

GwxDynamic is the object type from which dynamic GraphWorX objects (size dynamic, location dynamic, etc.) are derived. In other words, all dynamic GraphWorX objects have the properties and methods of **GwxDynamic**.

Properties

dataSource As String

The primary data source for the dynamic object. **DataSource** is a string which represents an OPC tag name, an expression, a constant value, or a GraphWorX local variable.

highRange As String

Represents the overridden high range for this dynamic. **HighRange** is a string which represents an OPC tag name, an expression, a constant value, or a GraphWorX local variable. This property is only used for dynamics based on an analog data source; it is ignored for dynamics which are based on digital connections.

lowRange As String

Represents the overridden low range for this dynamic. **LowRange** is a string which represents an OPC tag name, an expression, a constant value, or a GraphWorX local variable. This property is only used for dynamics based on an analog data source; it is ignored for dynamics which are based on digital connections.

objectName As String

Gets/Sets the object name of a dynamic object. The object name is used to identify the object when using certain OLE Automation methods (for instance **GwxDisplay::GetDynamicObjectFromName**). GraphWorX will insure that object names are unique. If you assign an object name that already exists for another dynamic object in a display, GraphWorX will append a number to the object name (for example, "size" would become "size1", "size1" would become "size2", etc.).

RangeOverride As Boolean

Gets/Sets range override status. When FALSE, GraphWorX will use the ranges associated with the primary **DataSource**. When TRUE, GraphWorX will use the ranges defined in the **HighRange** and **LowRange** properties. This property is only used for dynamics based on an analog data source (**GwxSize**, **GwxLocation**, **GwxRotation**, **GwxAnalogColor**, **GwxAnalogSelector**, **GwxProcessPoint**); it is ignored for dynamics which are based on digital connections.

RequestDataType As GWXDATATYPE

Allows you to specify the datatype used to request the data source from the OPC server.

TimerRate As Long

Frequency update rate for timer based dynamic types (this property is ignored for dynamics which are not timer based). Timer based dynamics include: **GwxFlash**, **GwxAnimator**, and **GwxPick**.

UserCustomData As String

This string is used to store custom data. Use this property to associate any additional data with the dynamic object.

UserDescription As String

A description string for the dynamic object. Typically, this string is used to be displayed as informational text in a tooltip.

Methods**Function GetAliasDefinition(aliasName As String) As String**

Gets the alias definition of the specified alias name.

Function GetDataSourcePointObject() As Object

Gets the **GwxPoint** object for the data source of this dynamic object.

Function GetHighRangePointObject() As Object

Gets the **GwxPoint** object for the high range of this dynamic object.

Function GetLowRangePointObject() As Object

Gets the **GwxPoint** object for the low range of this dynamic object.

Function GetNumberOfSubDynamics() As Long

Returns the number of immediate child sub-dynamic objects used in this dynamic object. Sub-dynamic objects are used by the dynamic types: **GwxDigitalColor**, **GwxDigitalSelector**, and **GwxPick**.

Function GetObjectTypeName() As String

Returns the type name of this object.

Function GetSubDynamicObjectFromIndex(index As Long) As Object

Gets the sub-dynamic object with the specified zero-based index, which is an immediate child of this dynamic object (index zero is the first child object). See also **GetNumberOfSubDynamics()**.

Function GetVisibleObject() As Object

Gets the visible object to which this dynamic is attached.

Function SetAliasDefinition(aliasName As String, newDefinition As String) As Boolean

Sets the alias definition for this dynamic object only. Returns FALSE if no matching alias names were found, TRUE otherwise. (See also **GwxDisplay::SetAliasDefinition** and **GwxVisible::SetAliasDefinition**).

Function SetAliases(commandString As String) As Boolean

Sets multiple alias definitions for this dynamic object. (See also, **GwxDisplay::SetAliases()** and **GwxDynamic::SetAliasDefinition()**).

Function SetAliasesFromFile(filename As String) As Boolean

Sets multiple alias definitions for this dynamic object. (See also, **GwxDisplay::SetAliasesFromFile()** and **GwxDynamic::SetAliasDefinition()**).

GwxDigitalSelector

The **GwxDigitalSelector** object is used to hide/show individual objects from a set of objects based on a digital (boolean) signal. **GwxDigitalSelector** objects are only associated with **GwxSymbol** objects (the **GwxSymbol** object defines the set of objects upon which the **GwxDigitalSelector** object acts). **GwxDigitalSelector** has all the properties and methods of **GwxDynamic**, plus the additional method described below.

Methods

Function SetConnectionInfo(objectNumber As Integer, dataSource As String, showWhenTrue As Boolean) As Boolean

Sets data connections for this digital selector object. The parameter **objectNumber** is a zero-based index representing an immediate child of this symbol object to which this dynamic is attached (index zero is the first child object). (See also **GwxDigitalSelectorInfo**).

GwxDigitalSelectorInfo

This object includes properties of individual data connections for **GwxDigitalSelector** objects. **GwxDigitalSelectorInfo** has all the properties and methods of **GwxDynamic**.

GwxDigitalColor

The **GwxDigitalColor** object is used to change the color of the associated visible object based on one or more digital (boolean) signals. **GwxDigitalColor** has all the properties and methods of **GwxDynamic**, plus the additional methods described below.

Methods

Function DeleteSubDynamic(index As Integer) As Boolean

This function deletes subdynamic object at the position index. It returns true on success. It returns false on failure, typically when the index is out of range.

Function SwapSubDynamics(index1 As Integer, index2 As Integer) As Boolean

This function swaps subdynamic objects at the positions index1 and index2. It returns true on success. It returns false on failure, typically when the index is out of range.

GwxDigitalColorInfo

The **GwxDigitalColorInfo** object includes properties of individual data connections for **GwxDigitalColor** objects. **GwxDigitalColorInfo** has all the properties and methods of **GwxDynamic**, plus the additional properties described below.

Properties

UseFillColor As Boolean

UseLineColor As Boolean

UseShadowColor As Boolean

These properties enable color changes on the visual object.

FillColor As OLE_COLOR

LineColor As OLE_COLOR

ShadowColor As OLE_COLOR

These properties define the colors for the color action.

ChangeColorOnTrue As Boolean

This property specifies if the action is fired when the OPC tag is evaluated to true or false.

GwxAnalogSelector

The **GwxAnalogSelector** object is used to hide/show individual objects from a set of objects based on an analog signal. **GwxAnalogSelector** objects are only associated with **GwxSymbol** objects (the **GwxSymbol** object defines the set of objects upon which the **GwxAnalogSelector** object acts). **GwxAnalogSelector** has all the properties and methods of **GwxDynamic**, plus the additional method described below.

Methods

Function SetSelectorRanges(rangeArray) As Boolean

Sets selector ranges for this dynamic. Returns TRUE for success, FALSE for failure.

The parameter **rangeArray** is a SAFEARRAY of floats. The number of elements in the array must equal the number of objects in the analog selector. Each index in the array corresponds to the index of an object in the analog selector. Each array element represents a percentage value for the high limit of that object's range. Each array element must be a value in the range 0 to 1. Each array element value must be greater than the previous element value. The last array element's value must always be 1.

GwxAnalogColor

The **GwxAnalogColor** object is used to change the color of the associated visible object based on an analog signal. **GwxAnalogColor** has all the properties and methods of **GwxDynamic**.

GwxAnimator

The **GwxAnimator** object is used to hide/show individual objects from a set of objects in a specified sequence, based on elapsed time. The **GwxAnimator** is activated/deactivated based on the value of a digital signal. **GwxAnimator** objects are only associated with **GwxSymbol** objects (the **GwxSymbol** object defines the set of objects upon which the **GwxAnimator** object acts). **GwxAnimator** has all the properties and methods of **GwxDynamic**.

GwxSize

The **GwxSize** object is used to change the size of the associated visible object based on the value of an analog signal. **GwxSize** has all the properties and methods of **GwxDynamic**.

GwxLocation

The **GwxLocation** object is used to change the location of the associated visible object based on the value of an analog signal. **GwxLocation** has all the properties and methods of **GwxDynamic**, plus the additional methods described below.

Methods

Sub GetCoordinateFromLocationDelta(offset As Single, cx, cy)

This is an advanced functionality added for the advanced user who wants to develop complex applications. General applications will not find a use for this function.

This function allows you to retrieve the coordinate that an object would have on a Location dynamic if the tag associated with the dynamic had a specific value (offset).

This method is very handy to determine the position that an object would have on a complex path such as a train track, starting from a known offset. It basically performs a transformation from a linear space (offset on the path) to a bidimensional space (coordinate on the display).

This method uses the same algorithm used for the location dynamic to determine the position of an object on the location path based on the tag value.

Using this method in combination with "setobjectdimension" it is possible to change the position of several objects in a display and relocate them on a track.

The minimum offset and maximum offset values are the same as the min and max range of the location dynamic.

Function GetLastEnteredValue()

Returns the last entered value, which might be different from the currently displayed value.

GwxRotation

The **GwxRotation** object is used to change the angle of the associated visible object based on the value of an analog signal. **GwxRotation** has all the properties and methods of **GwxDynamic**, plus the additional methods described below.

Methods

Function GetLastEnteredValue()

Returns the last entered value, which might be different from the currently displayed value.

GwxFlash

The **GwxFlash** object is used to toggle the color or visibility of the associated visible object at a specified rate. The **GwxFlash** object is activated/deactivated based on the value of a digital signal. **GwxFlash** has all the properties and methods of **GwxDynamic**.

GwxHide

The **GwxHide** object is used to toggle the visibility or enabled/disabled state of the associated visible object based on the value of a digital signal. The **GwxHide** has all the properties and methods of **GwxDynamic**.

GwxPick

Associating a **GwxPick** object with a visible object designates that the associated visible object will perform an active when clicked on by the operator during runtime mode. **GwxPick** objects can be associated with **GwxButton** objects to make fully functional pushbuttons. **GwxPick** has all the properties and methods of **GwxDynamic**.

Properties

aliasCommandString As String

If the **AliasCommandType** property is **AliasSetFromFile**, **AliasCommandString** should be a filename that specifies a text file containing tab-separated pairs of alias names and alias definitions (see also **SetAliasesFromFile()**). If **AliasCommandType** is **AliasSetDirect**, **AliasCommandString** should have the following format:

```
aliasName1<TAB>aliasDef1<CR-LF>aliasName2<TAB>aliasDef2<CR-LF> ... etc.
```

aliasCommandType As GwxALIASCOMMAND

AliasCommandType can be one of the following values:

```
AliasNoCommand      = 0
AliasSetFromFile    = 1
AliasSetDirect      = 2
```

NameParameter As String

The meaning of name parameter varies based on the value of the **PickAction** property.

If **PickAction** equals **PickLoadDisplay**, **PickDragDropLoad**, **PickPopupWindow**, or **PickEmbeddedWindow**, **NameParameter** is the Gwx display filename (.gdf).

If **PickAction** equals **PickLaunchApp**, **NameParameter** is an executable filename (.exe).

If **PickAction** equals **PickRunScript**, **NameParameter** is a VBA macro name.

If **PickAction** equals **PickSetAliases** or **PickAliasesDialog**, **NameParameter** is a visible object name.

pickAction As GwxPICKACTION

See **GwxDisplay::CreatePickDynamic** for description of this property.

ScriptText As String

GraphWorX 7.0 allows for associating VBScript and JScript with a pick action. The scripts associated with the pick action are executed when the pick event is triggered (for example, when the left mouse button is pressed). These scripts can be edited through the new integrated script toolbar or through the OLE automation interface. Each script can be written using VBScript or JScript language independently from the other scripts associated with the other pick action. This property allows for reading and writing the script source code currently associated with the pick action.

ScriptType As GWXSCRIPTTYPE

GraphWorX 7.0 allows for associating VBScript and JScript with a pick action. The scripts associated with the pick action are executed when the pick event is triggered (for example, when the left mouse button is pressed). These scripts can be edited through the new integrated script toolbar or through the OLE automation interface. Each script can be written using VBScript or JScript language independently from the other scripts associated with the other pick action. This property allows for getting and setting the script language currently associated with the pick action.

```
'VBA example, works only in configure mode
'example for creating pick action PickRunScript and set script text
and type

Dim o As Object
'creates rectangle
Set o = ThisDisplay.CreateRectangle( _
    50, 50, 100, 100, True, 100, 0, 2, _
    LineSolid, False, 0, EdgeNone, False, _
    "rectangle", False)
'adds pick action PickRunScript on the rectangle
Set o = ThisDisplay.CreatePickDynamic( _
    "rectangle", "dynamic", PickRunScript, _
    ButtonNormal, TriggerOnDown, MouseButtonLeft, _
    False, "", "", False, False, "", "", "")
'sets script properties - name of the script is
'derived from pick action name
o.ScriptType = ScriptTypeVBScript
o.ScriptText = "MsgBox " & Chr(34) & "Test OK" & Chr(34)
```

tracking As Boolean

If the pick dynamic is a toggle or download pick and a checkbox or radio button pick, setting tracking to TRUE will cause the button state to track the data value associated with this pick dynamic.

Methods**Sub SimulateClick(mouseButton As GWXMOUSEBUTTON)**

Causes this pick dynamic to behave as though the user clicked on this pick with the specified mouse button.

GwxProcessPoint

GwxProcessPoint objects are associated with **GwxText** objects to create alphanumeric display fields and data entry fields. A Process Point is used to show and/or download the value of an associated signal (analog, digital, or string). **GwxProcessPoint** has all the properties and methods of **GwxDynamic**.

Properties**ConfirmBeforeWrite As Boolean**

TRUE to prompt the user with a confirmation message before writing data values.

UseKeypad As Boolean

TRUE to use an onscreen number pad or keyboard for this data entry.

Methods

Function *GetLastEnteredValue()*

Returns the last entered value, which might be different from the currently displayed value.

GwxTimedate

GwxTimedate objects are associated with **GwxText** objects to create fields which display the current time and/or date. **GwxTimedate** has all the properties and methods of **GwxDynamic**.

GwxPoint

GwxPoint is the object type that GraphWorX uses to represent data connections. If several **GwxDynamic** objects are connected to the same data source, they reference a single shared **GwxPoint** object. **GwxPoint** objects handle OPC tags, expressions, constant values, and GraphWorX local variables. **GwxPoint** objects cannot be explicitly created or destroyed. GraphWorX automatically manages the lifetimes of **GwxPoint** objects based on the data source connections of the **GwxDynamic** objects in the display.

Properties

dataType As GWXDATATYPE

Data type of this point object. Valid values for **GWXDATATYPE** are:

<code>DataTypeShort</code>	= 2
<code>DataTypeLong</code>	= 3
<code>DataTypeFloat</code>	= 4
<code>DataTypeDouble</code>	= 5
<code>DataTypeString</code>	= 8
<code>DataTypeBool</code>	= 11
<code>DataTypeByte</code>	= 17

highRange As Variant

High range value associated with this point object.

lowRange As Variant

Low range value associated with this point object.

Value As Variant

Current data value of this point object. This property gets updated with new values during runtime mode.

Methods

Function *GetObjectTypeName()* As String

Returns the type name of this object.

Function *GetPointName()* As String

Gets the point name (data source string) of this point object.

Note:

Section 14

GraphWorX VBA Project

The GraphWorX VBA project is loaded whenever you launch the VBA Editor from the GraphWorX Application by pressing **ALT+F11**. This project contains groups of modules by default, such as:

- GraphWorX Native Objects
- Modules - *ThisDisplay* module and *GwxTools* module
- Forms

Each module can contain VBA code - functions, subroutines, event handlers, and global declarations.

GwxTools Module

GwxTools is a custom module with common subroutines used in the VBA Wizard. The *GwxTools* module is not available to the user until a VBA Wizard is used.

ThisDisplay Module

ThisDisplay is a special module representing current GraphWorX display.

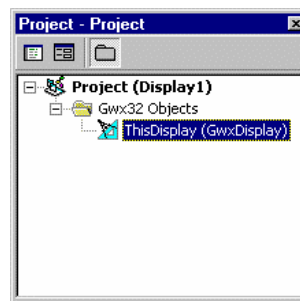


Figure 14.1. Project Window Displaying ThisDisplay Module

To display the code for *ThisDisplay* module, open the **Project Explorer** from the **View** menu of the VBA Editor. Expand the **GraphWorX Objects** group and double-click the *ThisDisplay (GwxDisplay)* item.

A *ThisDisplay* code window opens as below:

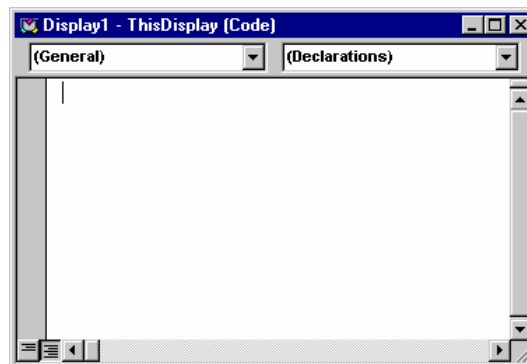


Figure 14.2. ThisDisplay Code Window

ThisDisplay contains two **combo boxes** at the top.

1. The top left combo box allows you to select items such as:
 - (General)
 - **GwxDisplay**

An empty display has only the above two items. If you have ActiveX controls in the display, the ActiveX items will also be displayed in this combo box.

2. The top right combo box shows events for the left combo selection. For example, the **GwxDisplay** item has events like **DisplayLoad**, **DisplayUnload**, and others.

Event Handling with the ThisDisplay Module

Event handlers can be added for a display using VBA. To do this, you should have the display loaded on the screen.

1. Open the VBA Editor by selecting **Macros > Visual Basic Editor** from the GraphWorX **Tools** menu. The GraphWorX VBA Project will be loaded by default.
2. Open the **ThisDisplay** module from the VBA Editor.
3. Select the **GwxDisplay** module in left combo box of **ThisDisplay** module.
4. Select the desired event from the list in the right combo box.
5. The subroutine header is automatically inserted to current module. Insert your VBA code to the body of the subroutine.
6. Save the GraphWorX Project by selecting **Save Display 1** from the **File** menu.
7. Exit the VBA Editor by selecting **Close and Return to Gwx** from the **File** menu.

Example

Write code for popping-up a message box which will display the message "Runtime Started" when you set your display into runtime.

1. Select the **GwxDisplay** module.
2. Select the **PostRuntimeStart** event from the list in the right combo box.
3. Insert VBA code, as shown in the figure below, to the body of the subroutine in the subroutine header:

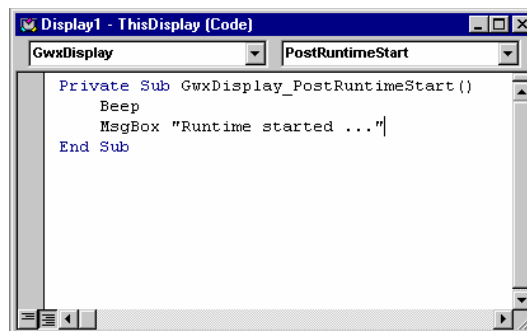


Figure 14.3. PostRuntimeStart Event Example

4. Switch to GraphWorX either by closing the VBA Editor or by pressing the **ALT+TAB** keys.
5. Test the event by starting runtime by clicking **Runtime** on the GraphWorX menu bar or by pressing the **CTRL+M** keys.
6. You should hear a beep, and a message box with a message "Runtime started" should appear.

How to Control GraphWorX Automation Using VBA

The VBA programmer has a full control over the properties and native objects of the current display and can control the rich animation interface of GraphWorX.

The Automation properties and methods are accessible through the *ThisDisplay* module. *ThisDisplay* is, as we said above, a representation of a current GraphWorX display and contains all Automation properties and methods.

Note: VBA can display Automation methods and properties only if the Automation object is checked in the **References** dialog, which can be opened from the menu bar. Select **References** from the VBA Editor **Tools** menu. Note that GraphWorX is checked automatically for every new display, as shown in the figure below.

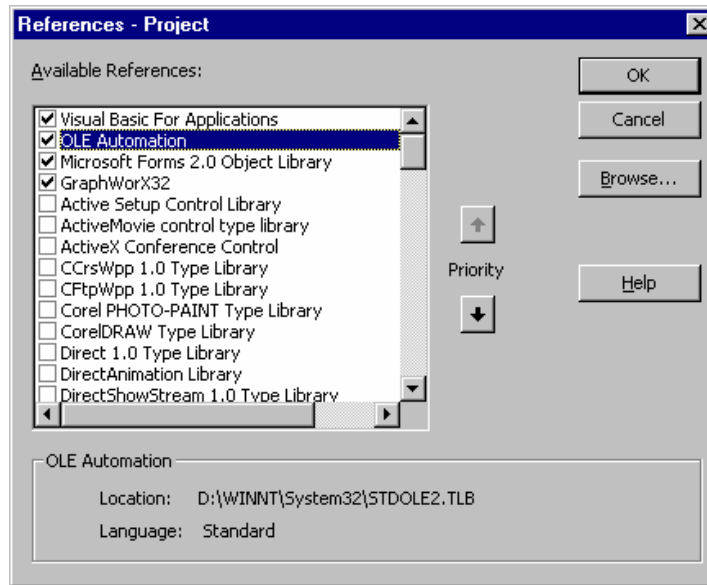


Figure 14.4. References Dialog

How to Access GraphWorX Native Display Objects

GraphWorX native objects like ellipses, rectangles, symbols and dynamic actions are not exposed to VBA by default. However they can be referenced from VBA if they have an *Object Name* assigned through the Property Inspector dialog.

A reference (actually a dispatch pointer) to a named object can be then retrieved by one of following methods of the ***ThisDisplay***, ***symbol***, and ***Visible*** objects.

- ***ThisDisplay*.GetDynamicObjectFromName**
- ***ThisDisplay*.GetVisibleObjectFromIndex**
- ***ThisDisplay*.GetVisibleObjectFromName**
- ***Visible*.GetDynamicObjectFromIndex**
- ***Visible*.GetDynamicObjectFromName**
- ***symbol*.GetVisibleObjectFromIndex**
- ***symbol*.GetVisibleObjectFromName**

***ThisDisplay*.GetVisibleObjectFromName("Name")**. This function takes the name of a visual object in the display and reports back the object associated with that name. Upon storing the result of this function in a variable, you can change the properties of the object in the display by changing the fields of the variable. The following example rotates a rectangle named "Square" 45 degrees from horizontal.

```
Dim obj As GwxRectangle
Set obj = ThisDisplay.GetVisibleObjectFromName("Square")
obj.Angle = 45
```

***ThisDisplay*.GetDynamicObjectFromName("Name")**. This function takes the name of a dynamic object in the display and reports back the dynamic object associated with that name. A dynamic object is not a physical, observable object but an abstraction for the operation performed on a visible object (ie. Hide, Rotation, Size). Once the dynamic object is stored in a variable, you can change its properties. The following example changes the data source of a GraphworX Hide object to zero, hiding the visible object associated with the dynamic object.

```
Dim obj As GwxHide
Set obj = ThisDisplay.GetDynamicObjectFromName("hd")
obj.dataSource = 0
```

ThisDisplay.GetVisibleObjectFromIndex(Long Value). This function selects an object based on the order in which visible objects on the screen were created. (The first visible object put on the screen has an index of 0) This function is useful for iterating through all the objects in a display. The following code turns the first object created to green and the second to red.

```
Dim obj1 As Object
Dim obj2 As Object
Set obj1 = ThisDisplay.GetVisibleObjectFromIndex(0)
Set obj2 = ThisDisplay.GetVisibleObjectFromIndex(1)
obj1.fillColor = RGB(0, 255, 0)
obj2.fillColor = RGB(255, 0, 0)
```

Symbolname.GetVisibleObjectFromIndex(Long value) Each GraphworX symbol has its own index that keeps track of the objects within it. The GetVisibleObjectFromIndex method, when appended to the name of a symbol, finds the visible object within the symbol with the specified index. This function is useful for iterating through all objects in a symbol. The following example turns the third visible object in a symbol named "sym" to green.

```
Dim sym1 As GwxSymbol
Dim obj As Object
Set sym1 = GetVisibleObjectFromName("sym")
Set obj = sym1.GetVisibleObjectFromIndex(2)
obj.fillColor = RGB(0,255,0)
```

(Note that the third object has index of 2. First has index 0)

Symbolname.GetVisibleObjectFromName("Name") Finds a visible object within a symbol by the object name specified in GraphworX.

VisibleObjectName.GetDynamicObjectFromIndex(Long value) Every time a dynamic object is assigned to a unique visible object, it is assigned an index. The first dynamic object assigned is given an index of zero. Since one visible object can be associated with many dynamic objects, this function provides a useful way of manipulating dynamic objects. The following code takes the second dynamic object associated with a rectangle named "rect" and changes its low range to 10.

```
Dim o_Vis As GwxRectangle
Dim o_Dyn As Object
Set o_Vis = ThisDisplay.GetVisibleObjectFromName("rect")
Set o_Dyn = o_Vis.GetDynamicObjectFromIndex(1)
o_dyn.lowRange = 10
```

VisibleObjectName.GetDynamicObjectFromName("Name") This function takes the name of a dynamic object associated with a visible object and allows you to represent the dynamic object with a variable.

Example

Write code for changing the color of an ellipse GraphWorX native object during Runtime.

1. Load an existing display or create a new GraphWorX display.
2. Draw an ellipse by clicking **Ellipse** on the **Draw** toolbar.
3. Double-click on the Ellipse. The **Property Inspector** dialog box opens, as shown in the figure below. Type the name **MyEllipse** in the object name field and then click on **OK**.
4. Open the VBA Editor by pressing **Alt+F11**.

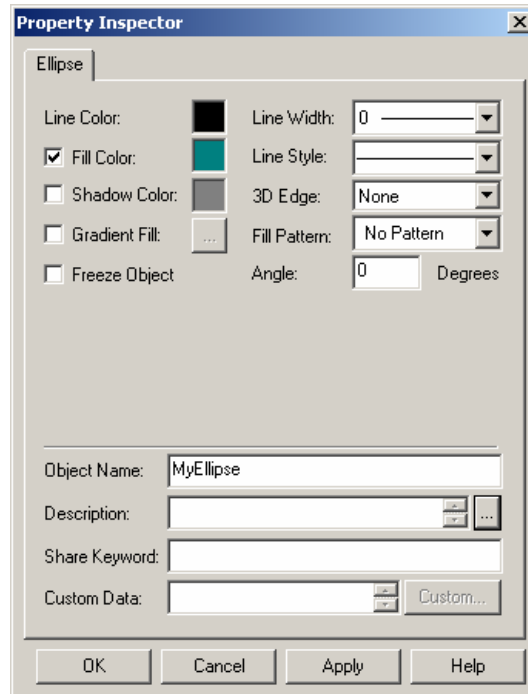


Figure 14.5. Ellipse Object

5. Select the **ThisDisplay** module.
6. Select the **GwxDisplay** module in left combo box of ThisDisplay module.
7. Select the **PostRuntimeStart** event from the list in the right combo box.
8. The subroutine header is automatically inserted into the current module. Insert the VBA code into the body of the subroutine, as shown in the figure below.

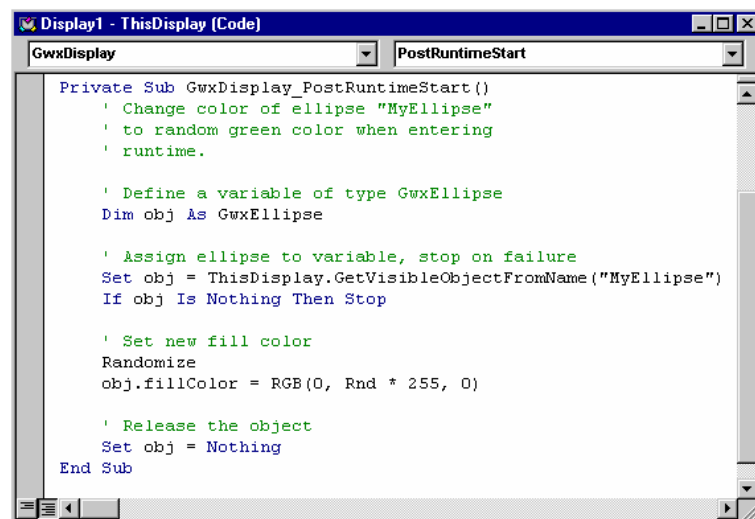


Figure 14.6. GraphWorX Ellipse Modified from VBA

9. Save the GraphWorX Project by selecting **Save Display 1** from the **File** menu.
10. Test the example by switching to the GraphWorX display and going into runtime.
11. The color of the ellipse should change to a random green color.

Object Names Must Be Unique

Sometimes it is desired to access the same objects in all duplicates (clones) of the desired symbol in the same way. This technique is useful, for example, for VBA Wizards, which have a macro behind the symbol. This macro is shared by all duplicates of the symbol and can be run on any of these duplicates.

To allow this feature, GraphWorX supports so called 'partial' names for objects in symbols. The partial name is a name that ends with an underscore '_', e.g. MyEllipse_. The duplicates of MyEllipse_ are then MyEllipse_1, MyEllipse_2, etc. What is important is the symbol method `GetVisibleObjectFromSymbol(partialName)`, which accepts this partial name and returns the first occurrence of the specified object in specified symbol.

Example

Assume we have a symbol that consists of a rectangle and an ellipse. We want to write code that modifies these objects in any copy of the symbol. We must assign partial names to both objects (let's name them MyRect_ and MyEll_). Here is the code that shows how to access these objects in a specific symbol:

```
Dim sym As GwxSymbol
Set sym = FindSomehowDesiredSymbol()
' user method to choose the symbol
Dim ell As GwxEllipse, rect As GwxRectangle
Set ell= sym.GetVisibleObjectFromName("MyRect_")
Set rect= sym.GetVisibleObjectFromName("MyEll_")
' do something with these objects
' release references
Set ell = nothing
Set rect = nothing
Set sym = nothing
```

Example 2

The following example demonstrates the above technique.

1. In GraphWorX, create one rectangle, and give it the Object Name **Rect_1**.
2. Duplicate this rectangle to create one more rectangle. Notice that this is automatically named **Rect_2**.
3. Group Rect_1 and Rect_2 together.
4. Duplicate this group using the duplicate button on the **Draw** toolbar.
5. Ungroup the duplicate object.
6. Read the Object name by using the Property Inspector dialog. Notice that the Object names of the two rectangles (the third and the fourth) are **Rect_3** and **Rect_4**.

Using VBA to Connect With Other Applications

One powerful feature of VBA is that it allows you to link to other Windows applications and to exchange data. In the following example, you will see how GraphWorX can send data to, and receive data from, an Excel worksheet by using VBA.

Example

Design a display and a spreadsheet, each with two data points, and have them communicate to each other through VBA.

1. Open a new GraphworX project and go to the VBA Editor from the *by* pressing **ALT+F11**.
2. The first thing you need to do is to get the VBA in GraphworX to recognize Microsoft Excel data types. In the Visual Basic Editor, select **References** from the **Tools** menu. This opens a list of available References to applications. Check the box next to "Microsoft Excel Object Library," as shown in the figure below.

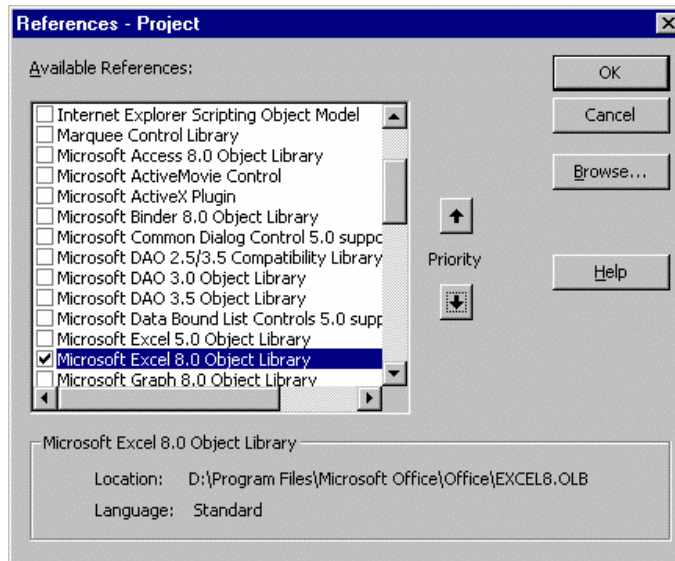


Figure 14.7. Selecting Microsoft Excel in the References Dialog

3. In the Project viewer window, double click on the *ThisDisplay* Module. You should see a code window with the words "Option Explicit." In that window, you want to make some global declarations that will be used later, as shown in the figure below:

Option Explicit

```
Public g_Excel_App As Excel.Application
Public g_Excel_Book As Excel.Workbook
Public g_Excel_Sheet As Excel.Worksheet
```

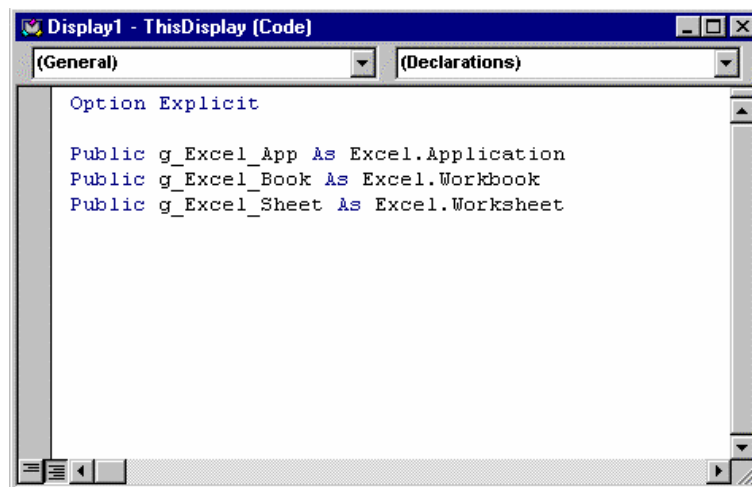


Figure 14.8. ThisDisplay Code Window with Global Declarations

4. In the left combo box at the top of the code window, select *GwxDisplay*. In the right combo box, select *PreRuntimeStart*. By entering the following code, you will cause Excel to be launched before GraphWorX goes into runtime.

```
Private Sub GwxDisplay_PreRuntimeStart()
    ' Open up Excel and make it visible
    Set g_Excel_App = CreateObject("Excel.application")
    g_Excel_App.Visible = True
    ' Open up a sheet
    Set g_Excel_Book = g_Excel_App.Workbooks.Add
    Set g_Excel_Sheet = g_Excel_Book.Worksheets(1)
    ' Initialize the two cells you will be using
    g_Excel_Sheet.Range("a1") = 0
    g_Excel_Sheet.Range("a2") = 0
End Sub
```

5. Create two process points in your GraphWorX display. Make sure, for both process points, that the **Data Entry** check box is checked, as shown in the figure below. Connect one to a local variable `~~a1~~` and the other to a local variable named `~~a2~~`. This double tilde notation is a standard notation for local variables in GraphWorX.

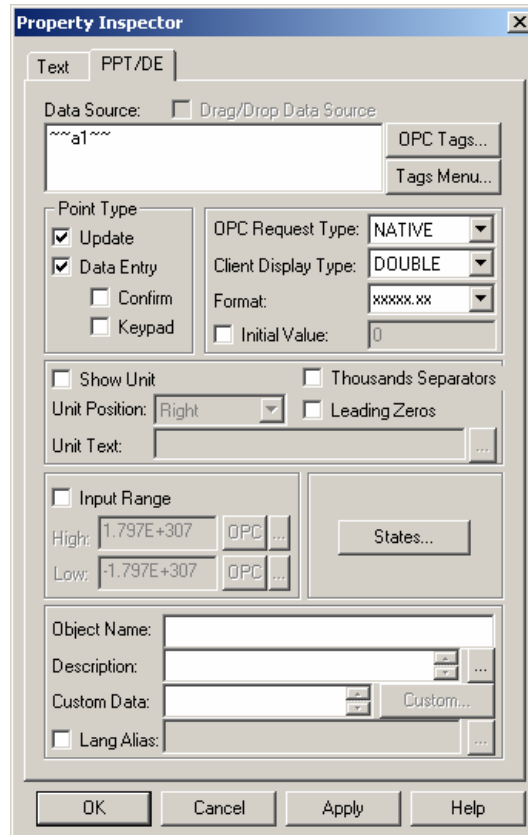


Figure 14.9. Configuration of Process Point

6. Now go back to the Visual Basic Editor. You will now make a new procedure designed to read from and write to Excel. Select **Procedure** from the **Insert** menu. Name the procedure "Read_Write" and then enter in the following code.

```
Public Sub Read_Write(ByVal Co As Integer)
' This procedure reads from or writes to excel
' based on the condition, Co, passed as a
' parameter

Dim Point As GwxPoint
Dim St As String 'Used to store cell name
Dim St2 As String 'Used to store variable name
Dim X As Integer
For X = 1 To 2
' add "a" to the value X converted to string
St = "a" + Mid(Str(X), 2, 1)
St2 = "~~" + St + "~~"
Set Point=ThisDisplay.GetPointObjectFromName(St2)
If Co = 1 Then 'Write to Excel
ThisDisplay.g_Excel_Sheet.Range(St)=Point.Value
Else 'Read From Excel
Point.Value=ThisDisplay.g_Excel_Sheet.Range(St)
End If
Next X
End Sub
```

7. Return to the GraphWorX display and create two radio buttons. Name one of them "Read from Excel" and the other "Write to Excel." Configure both of them to run a VBA Script. The execution trigger should be "while down." Create a new macro for each of the two radio buttons. Call one macro "Rd" and call the other "Wr." Both of these two macros will call the "Read_Write" procedure. The code for each macro should look something like this:

```
Sub Wr(o As GwxPick)
    Call ThisDisplay.Read_Write(1)
End Sub
```

```
Sub Rd(o As GwxPick)
    Call ThisDisplay.Read_Write(0)
End Sub
```

8. Go back to your GraphWorX display and go into runtime. You will notice that Excel starts up automatically with the value 0 in the "A1" and "A2" cells. If you click on the "Write to Excel" button and change the value of one of the process points in GraphWorX, you will notice the value in one of the cells in Excel changes. If you click on the "Read from Excel" button and change one of the two values in Excel, the process point will then update to match the value in Excel.

Other Sources of Information

For VBA programming, working with the modules and forms, and customization of the VBA Editor, refer to Microsoft Visual Basic Programmer's Guide or documentation on VBA and the help file that comes with VBA. You can open it from the VBA Editor **Help** menu.

Note that the VBA in GraphWorX is same as the one used in Microsoft Office applications (Word, PowerPoint, Access, Excel), and other products. Once you master VBA in GraphWorX, you are able to program in all other applications.

There are many examples included with GraphWorX, which are good sources of information and VBA programming tips and tricks.

It is also possible and helpful to open two or more instances of GraphWorX, in each instance open the VBA Editor, and copy and paste the VBA code between the instances.

For more information about VBA, please see the VBA Tutorial or the Microsoft VBA Help documentation.

Note:

GraphWorX Translator Utility

Introduction to the GraphWorX Translation Utility

The GraphWorX Translation Utility is a 32-bit Windows-based application, that takes a native 16-bit GraphWorX display or any text file using a script language as input and generates a native 32-bit GraphWorX display from it.

When converting a native 16-bit GraphWorX display, the translator uses a special utility that converts 16-bit displays into text files. Third-party applications can generate the text file providing the translator with the necessary input. The translator then uses the given text file and interprets it using GraphWorX Automation interface functions.

Converting Displays

Before you start using the translation utility, make sure that you have GraphWorX installed. The ProcessView installation provides GraphWorX. To open the GraphWorX Translation Utility from the Windows **Programs** menu, select **System302 > ProcessView > Tools > GFW16 to ProcessView**, as shown below.



Figure 1. Starting the GraphWorX Translation Utility

This opens the GraphWorX Translation Utility screen, shown below.

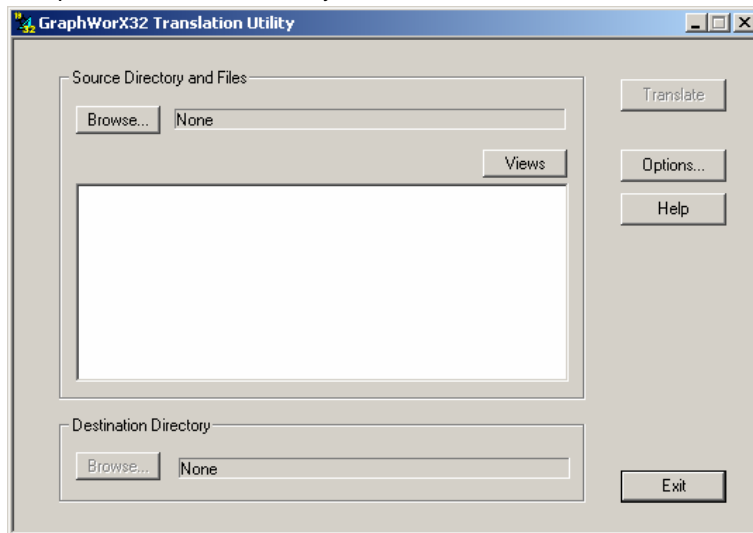


Figure 2. GraphWorX Translation Utility Screen

Native 16-bit GraphWorX displays

This section describes the steps required to translate native 16-bit GraphWorX displays to native 32-bit GraphWorX displays with GraphWorX Translator.

Defining Sources

After you start the application, the very first step is to select the source display files. Click on the **Browse** button in the **Source Directory and Files** section. This opens the **GraphWorX + 16-Bit Source Files** dialog box, shown below, which allows you to choose a *.gdf file. Use the Explorer capabilities to locate your 16-bit GraphWorX displays. Select the file, and then click **Open** button. All of the files you selected will be shown in the list box under the **Source Directory and Files** section.

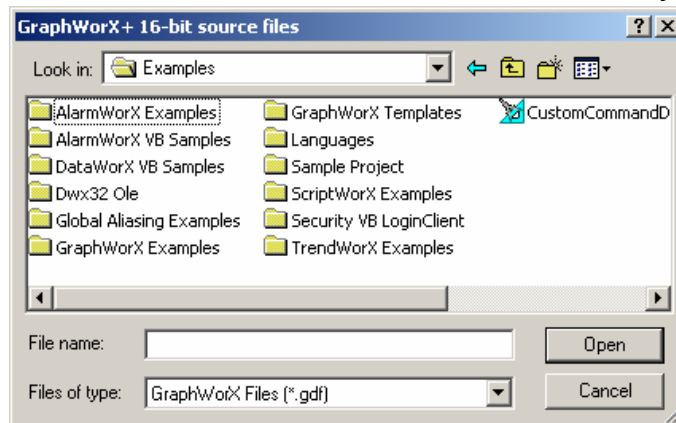


Figure 3. GraphWorX + 16-Bit Source Files Dialog Box

Defining Destination

The next step is to define the **Destination Directory**. Because 16-bit and 32-bit displays use the same extension, the same source and destination directory would cause the source files to be overridden by the new ones. Therefore, when defining the destination directory, be sure to use a different one from the source directory.

When you click the **Browse** button in the **Destination Directory** section, the **GraphWorX Destination Location** dialog box opens, as shown below. In this dialog box, your selected files are not used. Go into the desired location and click the **Ok** button. If accepted, the new destination location is shown in the **Destination Directory** section.

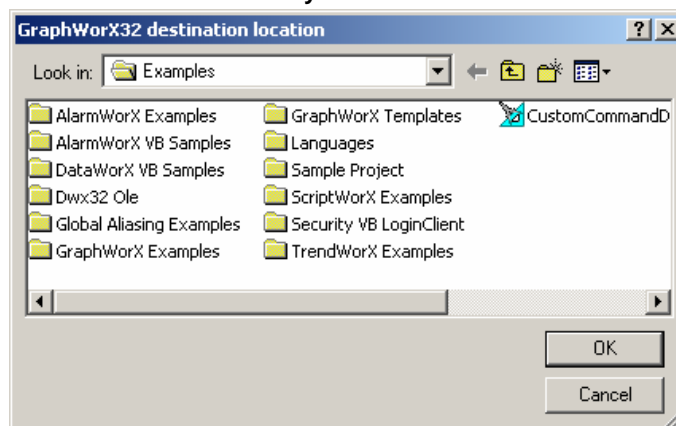


Figure 4. GraphWorX Destination Location Dialog Box

Translating

After both the source and the destination are chosen, the **Translate** button is enabled. Clicking this button will start the process of translating all the files, which are stored in the list box of **Source Directory and Files** section.

You are kept informed about the translation process by the progress bar. The progress ranges between 0 and 100 for each file. You can abort the translation by clicking the **Cancel** button. In this case, all the files created up to this point will be deleted, leaving the files and directories intact. No 32-bit displays will be generated.

If the installation fails, the progress bar will reset immediately and the next file will be processed. A Report file describing the problem will be generated in the destination directory. The intermediate ASCII text files generated by the 16-bit translation utility will be left in the source directory in order to localize the critical section. No 32-bit displays will be generated.

If the translation process succeeds, the Report file describing the translated display, including statistics, is again generated into the destination directory. As the translation finishes, it will show a common message for all the translated displays.

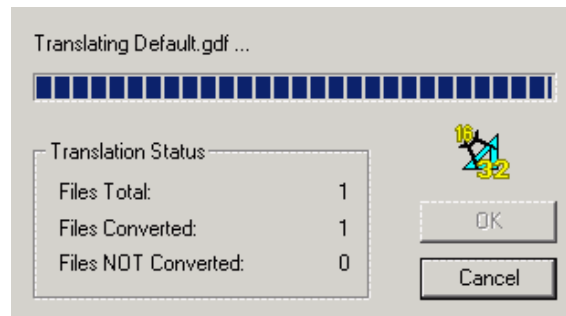


Figure 5. Translation

Using Options

You can override a few properties in GraphWorX for translation by clicking the **Options** button. This opens the **Options** dialog box.

The **Display** tab includes **Display Scaling** and **Display Dimensions** field. Generally, when you override any of the properties, the setting in the original display will not be considered. Your overridden setting will be accepted instead.

The **Window** tab contains a **Window Dimensions** field, which will set GraphWorX application window position and size.

If you want to change the source for the translator, go to the **General** tab and choose the desired source. The **Native GraphWorX** option will cause the translator to translate native 16-bit GraphWorX displays into 32-bit displays. When the **ASCII format file** option is chosen, you need to choose *.txt files as your source files. Each time you change the source, the translator will delete all the selected files (if any) in the list box of the **Source Directory and Files** section.

Third-Party Displays

This section describes the steps required to translate third-party displays to native 32-bit GraphWorX displays using GraphWorX Translation Utility. The sections below are more or less the same as for Native 16-bit display section (see above). Therefore, only the differences are described in the sections below.

Defining Sources for Third Party

Clicking the **Browse** button in the **Source Directory and Files** section displays the **ASCII Text Source Files** dialog box, shown below, which allows you to choose *.txt files as source files.

Note: It is assumed that the third-party application generated those text files prior to running the translator.

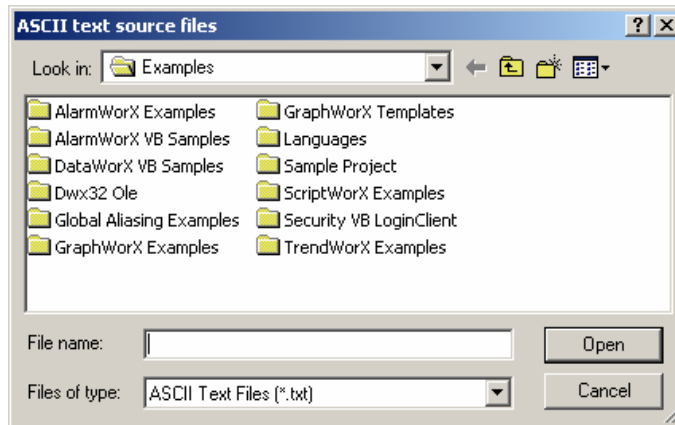


Figure 6. ASCII Text Source Files Dialog Box

Defining Destination for Third Party

To define the destination directory for the third party, click **Browse** in the **Destination Directory** section, just as you would for a native 16-bit GraphWorX display. This opens the **GraphWorX Destination Location** dialog box.

Translating for Third Party

The translation for a third party uses the same process as a translation for a 16-bit GraphWorX display, but no ASCII source files are removed regardless of the results of translation.

Using Options for Third Party

The options for a third party translation are the same as the options for 16-bit GraphWorX display translations.

Translator Controls

This section describes the Translator Controls.

Main Window

As you launch the application the following main window displays. You use this to operate the translator:

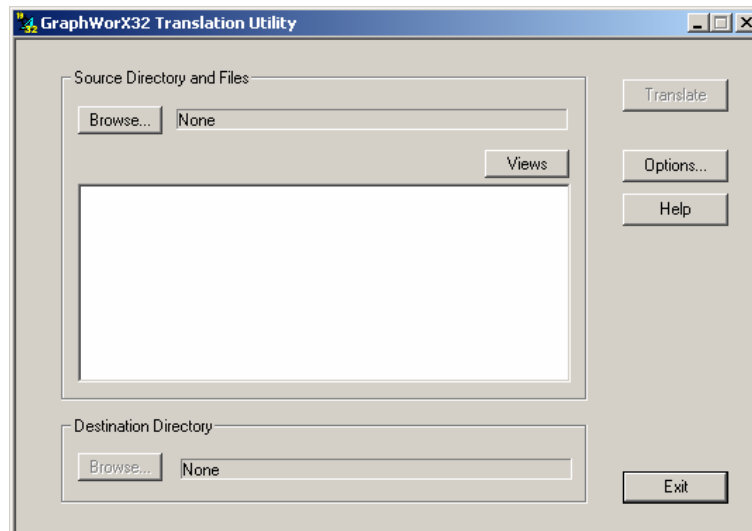


Figure 7. Main Window

Browsing Source Files

The first step is to choose the source files. By clicking the *Browse* button in the source files section you can view the following dialog enabling you to select the source files:

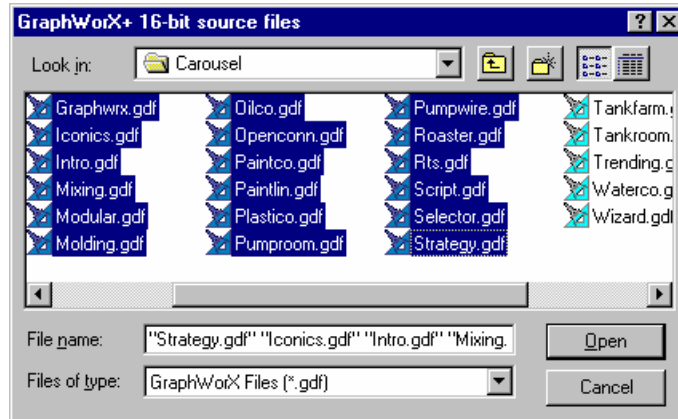


Figure 8. GraphWorX+ 16-Bit Source files Dialog

Depending on the source chosen (native or third-party), you can select either .gdf or .txt files.

Browsing the Destination Directory

As in the previous section you can define the destination directory using this dialog:

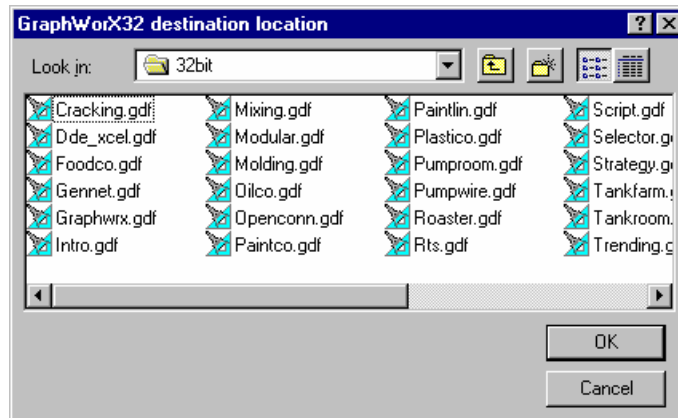


Figure 9. GraphWorX destination location Dialog

Note that the destination directory must differ from the source. This is to prevent the files from being overwritten.

Source File List Control

The **Views** button provides four different views on the files both translated and not translated. The four modes are:

List View

The list view shows all the files you selected for translation, as shown below.

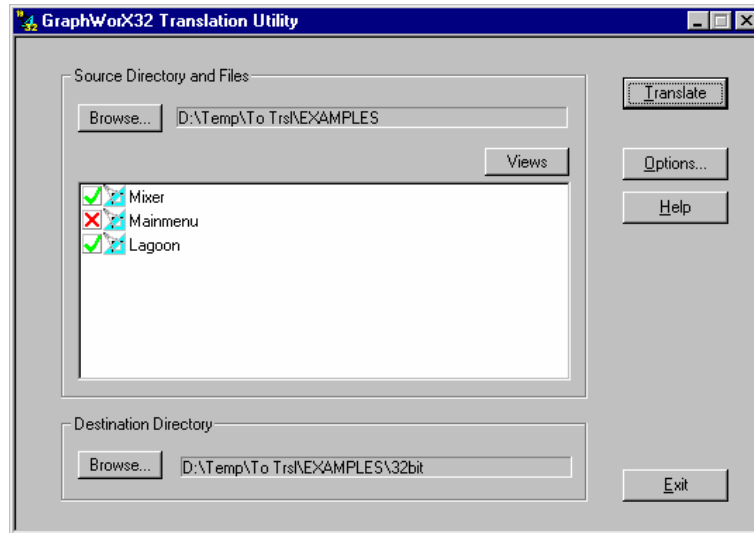


Figure 10. GraphWorX Translation Utility: List View

Report View

The Report view provides more detailed file information, including:

- **Name:** File name without extension.
- **File Size:** Source file size; if translated, the original/destination file size format is used.
- **Modified:** Source file modification information is shown.

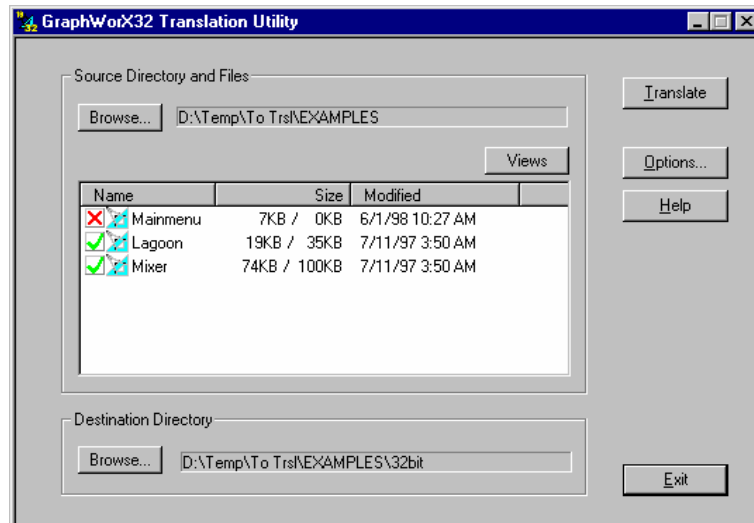


Figure 11. GraphWorX Translation Utility: Report View

Note: Clicking on the column headers sorts the column ascending or descending.

List View and List Box

If you click on one of the translated files, list box will show you the corresponding report file. Double-clicking on the file opens the notepad with the corresponding report file, as shown below.

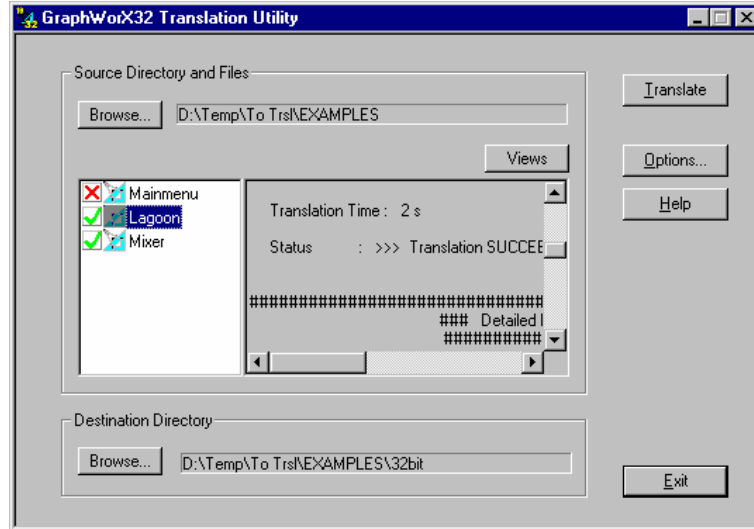


Figure 12. GraphWorX Translation Utility: List View and List Box

List View and GraphWorX Preview

When you click on one of the translated files, GraphWorX ActiveX will show a preview of the file, as shown below. You can also toggle runtime mode by clicking on the button that appears above the preview.

Double-clicking on the file opens the notepad with the corresponding report file.

Either in list or report view, when the translation process is done, there are two types of status icons (*succeeded*-green check mark and *failed*-red cross mark) shown in front of the items icons.

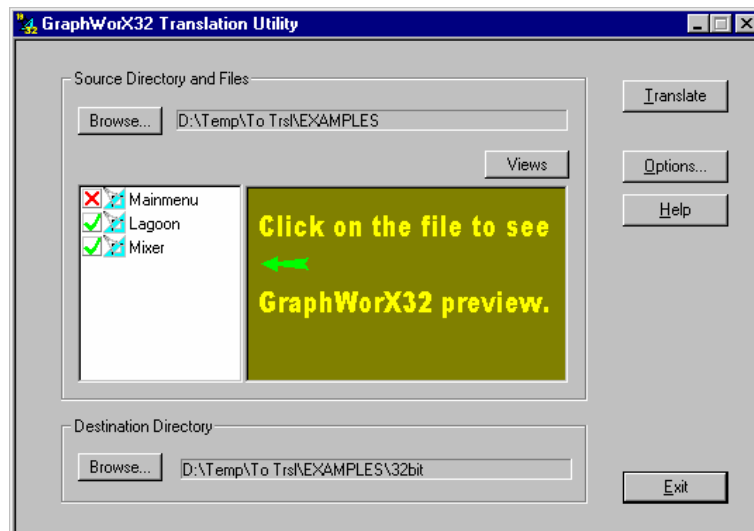


Figure 13. GraphWorX Translation Utility: List View and GraphWorX Preview

Option Controls

Option controls basically give you a method of overriding displays properties globally, i.e. for all translated displays.

Clicking the **Options** button on the main screen opens the Options dialog box, in which the following options can be modified:

General

The **Source** section of the **General** tab of the **Options** dialog box, shown below, has two items. When **Native GraphWorX** is selected, the translator will take 16-bit displays as a source. Otherwise, *.txt files will be taken.

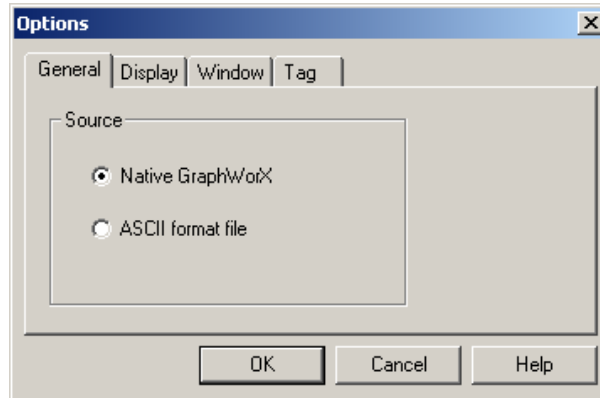


Figure 14. Options Dialog Box: General Tab

Display

In the **Display** tab of the **Options** dialog box, shown below, you can override **Display Scaling** and **Display Dimensions**. In that case, the setting in the source files will be ignored.

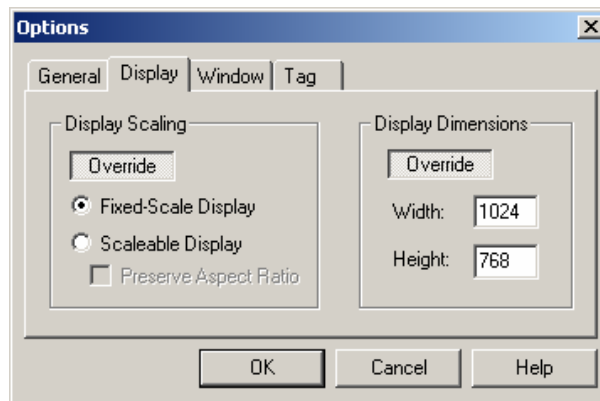


Figure 15. Options Dialog Box: Display Tab

Window

The **Window** tab of the **Options** dialog box, shown below, contains a **Window Dimensions** field, which will set the GraphWorX application window position and size.

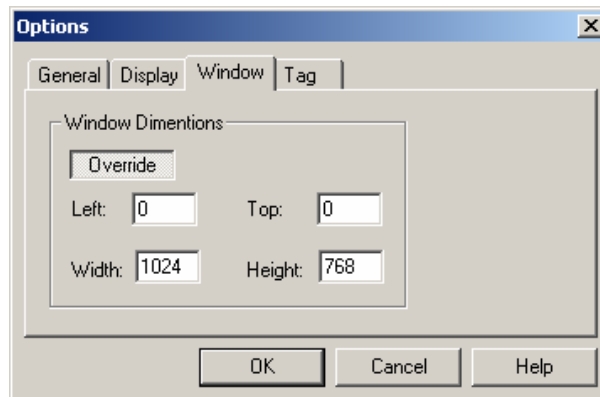


Figure 16. Options Dialog Box: Windows Tab

Tag

In the **Tag** tab of the **Options** dialog box, shown below, the **Prepend String** will be added in front of each tag in the display. The default was chosen in order to utilize a bridge between 16-bit and 32-bit software. However, you can easily define a different string to connect your tags to the desired field. For instance, instead of using Modbus I/O server, you can enter Modbus OPC Server.

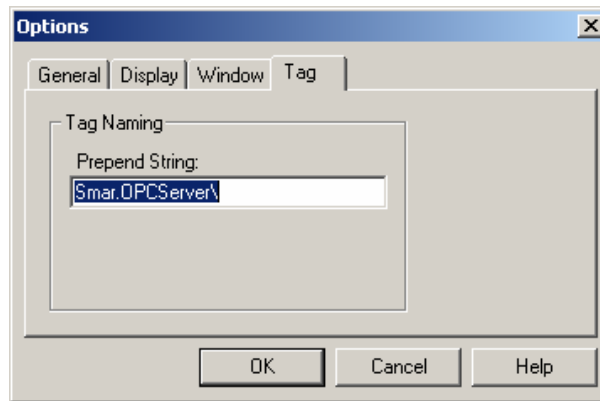


Figure 17. Options Dialog Box: Tag Tab

Script Language Reference

This section describes the Script Language Reference used in the GraphWorX Translation Utility.

OLE Automation is used to access the properties and methods of objects in GraphWorX.

Properties are used to reference attributes of an object (for example, a GraphWorX display has a *BackgroundColor* property). Methods are used to make the object perform an action (for example, a display has a *FileOpen* method, which loads a new display).

Properties and methods are called from GraphWorX Translation Utility as a response to ASCII text file either provided or generated.

There are numerous object types in GraphWorX, each with their own methods and properties. This section describes in detail, the objects supported by GraphWorX Translation Utility.

Generally, objects in GraphWorX can be placed in two groups: **visible** and **dynamic** objects.

Syntax

Script language is supposed to be used in a pure ASCII text file. Syntax for the language is:

Key: [param1, param2[, ...]]

Only one keyword per line is allowed. Line length limit is 255 characters. White spaces are ignored; commas separate parameters. Colon ending the key is compulsory. If all parameters are defined, the rest of the line is ignored and can be used as a comment.

There are two possibilities of how to define a string. First, as a regular character set without any commas (they are reserved as separators). Secondly, if commas needed, use quotation marks. If a string contains another quotation mark, you need to put the double one. The following are examples of valid strings:

Rectangle 2

"Rectangle1, 2 and 3"

"This symbol stands for: ""A negation""."

The following keywords are defined for the purpose of using the Translator:

- **Keyword Keys**

REM, BEGIN, END, SYMBOL, END_SYMBOL

- **Window Keys**
WINDOW_PIXELS, WINDOW_PERCENT, VIEW, DISPLAY_SIZE, DISPLAY_SCALE_MODE, DISPLAY_COLOR,
- **Visible Keys**
ELLIPSE, RECTANGLE, TEXT, POLYLINE, ARC, BUTTON
- **Dynamic Keys**
SIZE, LOCATION, ROTATION, HIDE, FLASH, ANALOG_COLOR, DIGITAL_COLOR, PICK, TIMEDATE, PROCESS_POINT, ANIMATOR, ANALOG_SELECTOR, DIGITAL_SELECTOR

Keyword Keys

The following keys do not require any parameters.

REM

Serves for your comments. Whole line is ignored.

BEGIN

This key closes the section for defining *Window* and *Display* (see *Window Keys*) and opens the section for defining visual and dynamic objects. Only keys from *Visible* and *Dynamic Keys* can be used after BEGIN key.

END

This key closes the whole script. Any other lines are ignored.

SYMBOL

Starts logging visible objects to be part of the symbol.

END_SYMBOL

Closes symbol definition. There must be at least one visible object defined between SYMBOL and END_SYMBOL keys. There are two parameters following this keyword:

BSTR UserDescription, BSTR UserCustomData.

WARNING

This is a supplementary keyword only. When the translator encounters such a keyword, it queues it and writes all the warnings together into the report file.

There is one parameter following this keyword: **BSTR wrnText.**

Window Keys

DISPLAY_COLOR

Parameters: **OLE_COLOR BackgroundColor**

Gets/Sets the current display's background color.

DISPLAY_SCALE_MODE

Parameters: **short ScaleMode**

Sets the scaling mode of the current display. Valid values are:

FixedScale	= 0
Scaleable	= 1
ScaleablePreserveAspect	= 2

WINDOW_PIXELS

Parameters: **long left, long top, long width, long height**

Sets the GraphWorX main window size and location in pixels.

WINDOW_PERCENT

Parameters: **float left, float top, float width, float height**

Sets the GraphWorX main window size and location as a percentage of the total screen size. Parameters should be values in the range of 0.0 to 1.0.

DISPLAY_SIZE

Parameters: **long width, long height**

Sets the display dimensions (work area/world bounds) of the currently loaded display.

VIEW

Parameters: **long left, long top, long width, long height**

Sets the GraphWorX view rectangle's size and location. The view dimensions define what portion of the work area/whole display is visible. Setting the view dimensions can be used to zoom and pan the view of a display.

Visible Keys**ELLIPSE**

Parameters Line1: **float left, float top, float width, float height, boolean isFilled, OLE_COLOR fillColor, OLE_COLOR lineColor, long lineWidth, GWXLINestyle lineStyle, boolean hasShadow, OLE_COLOR shadowColor, GWX3DEDGESTYLE edgeStyle, boolean isHidden, BSTR objectName**

Parameters Line2: **BSTR UserDescription, BSTR UserCustomData**

Ellipse

Has all the properties and methods of **Visible**.

RECTANGLE

Parameters Line1: **float left, float top, float width, float height, boolean isFilled, OLE_COLOR fillColor, OLE_COLOR lineColor, long lineWidth, GWXLINestyle lineStyle, boolean hasShadow, OLE_COLOR shadowColor, GWX3DEDGESTYLE edgeStyle, boolean isHidden, BSTR objectName, boolean rounded**

Parameters Line2: Same as for ELLIPSE object.

Rectangle

Has all the properties and methods of **Visible**, plus the additional properties described below.

boolean Rounded

When getting this property, if the value is TRUE, the rectangle has rounded corners. When setting this property to TRUE, the rectangle is given rounded corners with the default rounding settings.

long RoundingX

Gets/sets the amount of horizontal rounding of the rectangle's corners.

long RoundingY

Gets/sets the amount of vertical rounding of the rectangle's corners.

TEXT

Parameters Line1: **float x, float y, float w, float h, BSTR text, long alignment, boolean stretchText, boolean isFilled, OLE_COLOR fillColor, OLE_COLOR lineColor, long lineWidth, GWXLINestyle lineStyle, boolean hasShadow, OLE_COLOR shadowColor, GWX3DEDGESTYLE edgeStyle, boolean isHidden, BSTR objectName**

Parameters Line2: Same as for ELLIPSE object.

Parameters Line3: **long lfHeight, long lfWidth, long lfEscapement, long lfOrientation, long lfWeight, byte lfItalic, byte lfUnderline, byte lfStrikeOut, byte**

IfCharSet, byte IfOutPrecision, byte IfClipPrecision, byte IfQuality, byte IfPitchAndFamily, BSTR IfFaceName

Text

Has all the properties and methods of **Visible**, plus the additional properties described below.

float w, float h

Width and height are ignored if negative.

BSTR Text

This string is the text that is displayed by the text object.

boolean StretchText

TRUE if the font size should be resized when the text object is stretched. FALSE if the font size should remain the same when the object is stretched.

GWXTEXTALIGNMENT Alignment

The alignment of the text (left, center, right). This property is only significant for multiline text strings. Valid values for **GWXTEXTALIGNMENT** are:

TextAlignLeft	= 0
TextAlignCenter	= 1
TextAlignRight	= 2

Second parameter line represents **LOGFONT** system font structure.

POLYLINE

Parameters Line1: **boolean isFilled, OLE_COLOR fillColor, OLE_COLOR lineColor, long lineWidth, GWXLINESTYLE lineStyle, boolean hasShadow, OLE_COLOR shadowColor, GWX3DEDGESTYLE edgeStyle, boolean isHidden, BSTR objectName**

Parameters Line2: **long count**

Parameters Line3: **float x1, float y1**

Parameters Linecount+2: **float xcount, float ycount**

Parameters Linelast: Same as parameter line2 for ELLIPSE object.

Polyline

Has all the properties and methods of **Visible**.

ARC

Parameters Line1: **GWXARCTYPE arcType, float centerX, float centerY, float radiusX, float radiusY, float startAngle, float endAngle, boolean isFilled, OLE_COLOR fillColor, OLE_COLOR lineColor, long lineWidth, GWXLINESTYLE lineStyle, boolean hasShadow, OLE_COLOR shadowColor, GWX3DEDGESTYLE edgeStyle, boolean isHidden, BSTR objectName**

Parameters Line2: Same as for ELLIPSE object.

Arc

Has all the properties and methods of **Visible**, plus the additional properties described below.

GWXARCTYPE ArcType

Gets/sets the arc type of the arc. Valid values for **GWXARCTYPE** are:

ArcArc	= 0
ArcPie	= 1
ArcChord	= 2

float StartAngle

Gets/sets the start angle of the arc (in degrees).

float EndAngle

Gets/sets the end angle of the arc (in degrees).

BUTTON

Parameters Line1: **long buttonType, float x, float y, float w, float h, BSTR label, GWXTEXTALIGNMENT alignment, boolean stretchText, boolean isFilled, long fillColor, long lineColor, long lineWidth, long lineStyle, boolean hasShadow, long shadowColor, long edgeStyle, boolean isHidden, BSTR objectName**

Parameters Line2: Same as for ELLIPSE object.

Button

Has all the properties and methods of **Visible** and **Text**.

BITMAP

Parameters Line1: **BSTR fileName, float left, float top, float width, float height, boolean isFilled, OLE_COLOR fillColor, OLE_COLOR lineColor, long lineWidth, GWXLINESTYLE lineStyle, boolean hasShadow, OLE_COLOR shadowColor, GWX3DEGESTYLE edgeStyle, boolean isHidden, BSTR objectName**

Parameters Line2: Same as for ELLIPSE object.

If the **width** and **height** are negative or equal to zero, the bitmap is created in default size given by bitmap pixels.

FileName

Specifies the bitmap file name without any path. The translator will add the path of the text metafile and will remove the file immediately after creating the bitmap object.

METAFILE

Parameters Line1: **BSTR fileName, float left, float top, float width, float height, BSTR objectName**

Parameters Line2: Same as for ELLIPSE object.

FileName

Specifies the metafile name without any path. The translator will add the path of the text metafile file and will remove the file immediately after creating the metafile object.

Visible

Visible is the object type from which visible GraphWorX objects (ellipses, rectangles, etc.) are derived. In other words, all visible GraphWorX objects have the properties and methods of **Visible**.

boolean Visible

TRUE if the object is visible, FALSE if the object is hidden.

OLE_COLOR FillColor

Gets/Sets the visible object's fill color.

boolean IsFilled

TRUE if the object is filled, FALSE if the object is not filled.

OLE_COLOR LineColor

Gets/Sets the visible object's line/border color.

OLE_COLOR ShadowColor

Gets/Sets the visible object's shadow color.

boolean HasShadow

TRUE if the object has a shadow, FALSE if the object does not have a shadow.

long LineWidth

The width of an object's line/border. This value must be in the range of 0 to 10.

GWXLINestyle LineStyle

The style of the object's line/border. Valid values for GWXLINestyle are:

LineSolid = 0
LineDash = 1
LineDot = 2
LineDashDot = 3
LineDashDotDot = 4
LineNone = 5

GWX3DEDGESTYLE EdgeStyle

The 3-D edge style of the object's border. Valid values for GWX3DEDGESTYLE are:

EdgeNone = 0
EdgeRaised = 5
EdgeEtched = 6
EdgeBump = 9
EdgeSunken = 10

boolean Selected

TRUE if the object is selected, FALSE if the object is not selected. Selected objects are objects that have resize grips around them. *This property can only be changed in configure mode.*

float Angle

Gets/Sets the rotation angle of the object. The angle is specified in degrees.

BSTR ObjectName

Gets/Sets the object name of a visible object. The object name is used to identify the object when using certain OLE Automation methods (for instance Display::GetVisibleObjectFromName). GraphWorX will ensure that object names are unique. If you assign an object name that already exists for another visible object in a display, GraphWorX will append a number to the object name (for example, "tank" would become "tank1", "tank1" would become "tank2", etc.).

Dynamic Keys

Dynamic objects can be attached to any visible objects (see exceptions below). In scripting terminology, if the visible object has any dynamic actions attached, then they have to be specified after defining visible object, in the following line.

Many dynamic objects can be attached to one visible object. Also, dynamic repetitions are allowed. For instance one rectangle can hold 4 size actions and 3 rotations.

SIZE

Parameters Line1: **BSTR dynamicObjectName, GWXSIZEDYNTYPE sizeType, boolean clip, float startSize, float endSize**

Parameters Line2: **BSTR DataSource, boolean RangeOverride, BSTR HighRange, BSTR LowRange, BSTR UserDescription, BSTR UserCustomData**

Creates a **GwxSize** object with the specified attributes and attaches it to the visible object with the specified object name. The parameters **startSize** and **endSize** are percentage values in the range of 0.0 to 1.0. Valid value for **GWXSIZEDYNTYPE** are:

SizeLeft	= 0
SizeRight	= 1
SizeUp	= 2
SizeDown	= 3
SizeUpLeft	= 4
SizeUpRight	= 5
SizeDownLeft	= 6
SizeDownRight	= 7
SizeLeftRight	= 8
SizeUpDown	= 9
SizeLeftRightBias	= 10
SizeUpDownBias	= 11
SizeAllFour	= 12
SizeLeftRightUp	= 13
SizeLeftRightDown	= 14
SizeUpDownLeft	= 15
SizeUpDownRight	= 16

Size

Has all the properties and methods of **Dynamic**.

LOCATION

Parameters Line1: **BSTR dynamicObjectName, float offsetX, float offsetY, boolean slider, boolean tracking, short numberOfDetents, boolean continuousUpdate**

Parameters Line2: Same as for SIZE object

Location

Has all the properties and methods of **Dynamic**.

ROTATION

Parameters Line1: **BSTR dynamicObjectName, float startAngle, float endAngle, float pivotX, float pivotY, boolean clockwise, boolean dial, boolean tracking, short numberOfDetents, boolean continuousUpdate**

Parameters Line2: Same as for SIZE object

Rotation

Has all the properties and methods of **Dynamic**.

HIDE

Parameters Line1: **BSTR dynamicObjectName, boolean hideWhenTrue, boolean disableObject**

Parameters Line2: Same as for SIZE object

Hide

Has all the properties and methods of **Dynamic**.

FLASH

Parameters Line1: **BSTR dynamicObjectName, boolean hideObject, boolean flashWhenTrue, boolean altStateWhenOff, boolean changeFill, boolean changeLine, boolean changeShadow, OLE_COLOR altFillColor, OLE_COLOR altLineColor, OLE_COLOR altShadowColor**

Parameters Line2: **BSTR DataSource, long TimerRate, BSTR UserDescription, BSTR UserCustomData**

Flash

Has all the properties and methods of Dynamic.

ANALOG_COLOR

Parameters Line1: **BSTR dynamicObjectName, boolean changeFill, boolean changeLine, boolean changeShadow, OLE_COLOR startFillColor, OLE_COLOR endFillColor, OLE_COLOR startLineColor, OLE_COLOR endLineColor, OLE_COLOR startShadowColor, OLE_COLOR endShadowColor, boolean defaultColorAbove, boolean defaultColorBelow**

Parameters Line2: Same as for SIZE object

AnalogColor

Has all the properties and methods of Dynamic.

DIGITAL_COLOR

Parameters Line1: **BSTR dynamicObjectName, boolean changeColorWhenTrue, boolean changeFill, boolean changeLine, boolean changeShadow, long fillColor, long lineColor, long shadowColor, BSTR dataSource**

Parameters Line2: Same as for TIMEDATE object

DigitalColor

Has all the properties and methods of Dynamic.

There can be only one **DigitalColor** dynamic attached to a visible. However, such a dynamic can hold many data sources as individual inputs to alternate color. To add another **DigitalColor** to the visible, create (call) another **DigitalColor** object with different data source.

Parameter line two is common for all the DigitalColors. The last DigitalColor defined will be taken into effect.

PICK

Parameters Line1: **BSTR dynamicObjectName, GWXPICKACTION pickAction, GWXBUTTONTYPE pickType, GWXEXECUTIONTRIGGER executionTrigger, GWXMOUSEBUTTON mouseButton, boolean initiallySelected, BSTR groupName, BSTR fileName, boolean modal, boolean center, BSTR value1, BSTR value2, BSTR value3**

Parameters Line2: Same as for FLASH object

Valid values for **GWXPICKACTION** are:

PickLoadDisplay	= 0
PickDragDropLoad	= 1
PickPopupWindow	= 2
PickDownloadValue	= 3
PickToggleValue	= 4
PickLaunchApp	= 5
PickClose	= 6
PickRunScript	= 7

Valid values for **GWXBUTTONTYPE** are:

ButtonNormal	= 0
ButtonCheck	= 1
ButtonRadio	= 2

Valid values for **GWXEXECUTIONTRIGGER** are:

TriggerOnDown	= 1
TriggerWhileDown	= 2
TriggerOnDnWhileDn	= 3
TriggerOnUp	= 4
TriggerOnDnOnUp	= 5
TriggerWhileDnOnUp	= 6
TriggerOnDnWhileDnOnUp	= 7

Valid values for GWXMOUSEBUTTON are:

```

MouseButtonLeft    = 0
MouseButtonMiddle  = 1
MouseButtonRight   = 2

```

Pick has all the properties and methods of **Dynamic**.

TIMEDATE

Note: This dynamic object can be attached to text object only!

Parameters Line1: **BSTR dynamicObjectName, long formatType, BSTR timeFormat, BSTR dateFormat**

Parameters Line2: **BSTR UserDescription, BSTR UserCustomData**

Timedate

Has all the properties and methods of **Dynamic**.

PROCESS_POINT

Note: This dynamic object can be attached to *text* object only!

Parameters Line1: **BSTR dynamicObjectName, GWXDATATYPE dataType, boolean update, boolean dataEntry, boolean hasInitialValue, VARIANT initialValue, BSTR format**

Parameters Line2: Same as for **SIZE** object

ProcessPoint

Has all the properties and methods of **Dynamic**.

ANIMATOR

Note: This dynamic object can be attached to *symbol* object only!

Parameters Line1: **BSTR dynamicObjectName, boolean animateWhenTrue, boolean visibleWhenOff, boolean currentFrameWhenOff**

Parameters Line2: Same as for **FLASH** object

Animator

Has all the properties and methods of **Dynamic**.

ANALOG_SELECTOR

Note: This dynamic object can be attached to *symbol* object only!

Parameters Line1: **BSTR dynamicObjectName, boolean hiddenWhenAbove, boolean hiddenWhenBelow**

Parameters Line2: Same as for **SIZE** object

AnalogSelector

Has all the properties and methods of **Dynamic**.

DIGITAL_SELECTOR

Note: This dynamic object can be attached to *symbol* object only!

Parameters Line1: **BSTR dynamicObjectName;**

Parameters Line2: Same as for **TIMEDATE** object

Parameters Line3 : **long objCount**

Parameters Line4 : **short objectNumber, LPCTSTR dataSource, BOOL showWhenTrue**

Parameters Line objCount+3 : **short objectNumber, LPCTSTR dataSource, BOOL showWhenTrue**

DigitalSelector

Has all the properties and methods of **Dynamic**, plus the additional methods described below.

Point

Is the object type, which GraphWorX uses to represent data connections. If several **Dynamic** objects are connected to the same data source, they reference a single shared **Point** object.

Point objects handle OPC tags, expressions, constant values, and GraphWorX local variables. **Point** objects cannot be explicitly created or destroyed. GraphWorX automatically manages the lifetimes of **Point** objects based on the data source connections of the **Dynamic** objects in the display.

VARIANT Value

Current data value of this point object. This property gets updated with new values during runtime mode.

VARIANT HighRange

High range value associated with this point object.

VARIANT LowRange

Low range value associated with this point object.

GWXDATATYPE DataType

Data type of this point object. Valid values for GWXDATATYPE are:

DataTypeShort	= 2
DataTypeLong	= 3
DataTypeFloat	= 4
DataTypeDouble	= 5
DataTypeString	= 8
DataTypeBool	= 11
DataTypeByte	= 17

Dynamic

Dynamic is the object type from which dynamic GraphWorX objects (size dynamic, location dynamic, etc.) are derived. In other words, all dynamic GraphWorX objects have the properties and methods of **Dynamic**.

BSTR DataSource

The primary data source for the dynamic object. **DataSource** is a string which represents an OPC tag name, an expression, a constant value, or a GraphWorX local variable.

boolean RangeOverride

Gets/Sets range override status. When FALSE, GraphWorX will use the ranges associated with the primary **DataSource**. When TRUE, GraphWorX will use the ranges defined in the **HighRange** and **LowRange** properties. This property is only used for dynamics based on an analog data source (**Size**, **Location**, **Rotation**, **AnalogColor**, **AnalogSelector**, **ProcessPoint**); it is ignored for dynamics which are based on digital connections.

BSTR HighRange

Represents the overridden high range for this dynamic. **HighRange** is a string which represents an OPC tag name, an expression, a constant value, or a GraphWorX local variable. This property is only used for dynamics based on an analog data source; it is ignored for dynamics that are based on digital connections.

BSTR LowRange

Represents the overridden low range for this dynamic. **LowRange** is a string which represents an OPC tag name, an expression, a constant value, or a GraphWorX local variable. This property is only used for dynamics based on an analog data source; it is ignored for dynamics which are based on digital connections.

long TimerRate

Frequency update rate for timer based dynamic types (this property is ignored for dynamics which are not timer based). Timer based dynamics include: **Flash**, **Animator**, and **Pick**.

BSTR UserDescription

A description string for the dynamic object. Typically, this string is used to be displayed as informational text in a tooltip.

BSTR UserCustomData

This string is used to store custom data. Use this property to associate any additional data with the dynamic object.

FIX to GWX Translation Utility

Introduction

ProcessView allows you to convert from Intellution FIX displays to GraphWorX displays. The "FixToGwxTranslator.exe" application, located in the **Tools** directory of the ProcessView product CD, is a simple utility that converts FIX *.jdf files to GraphWorX *.gdf files.

What Are .jdf Files?

The *.jdf files are produced by a "FIX Web Server conversion utility", which is available as a part of the FIX Web Server installation. This utility will convert FIX picture files (.odf) to Web Server files (.jdf, .bmp).

Using the Translator

To convert FIX picture files, start the "FixToGwxTranslator.exe" from the **Tools** directory on the ProcessView product CD. This opens the user interface, shown below. First select the source *.jdf file by clicking the **Browse** button (top). The *.jdf files are produced by a "Fix Web server conversion utility" (see the **FIX Web Server Conversion Utility** section below).

Then select the destination *.gdf file to which the source file will be converted by clicking the **Browse** button (bottom). By default, the ProcessView Working Directory is preset in the edit box. The ProcessView working directory can be changed using the GraphWorX application by selecting **Set Working Directory** from the **Tools** menu.

Then click the **Translate** button. To see and check the results, use GraphWorX or the GraphWorX Viewer ActiveX.

The FixToGwxTranslator utility is capable of converting only one FIX display at a time. For conversion of all the FIX displays, run the utility repetitively.

Note: Translating of Intellution pictures that contain thousands of objects is time consuming.

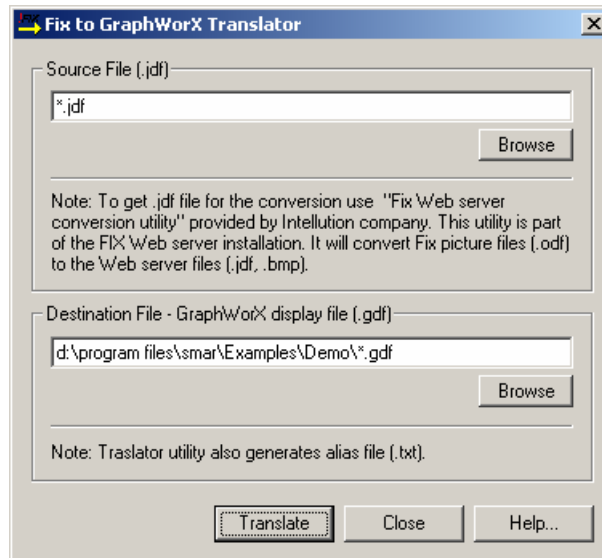


Figure 1. FIX to GraphWorX Translation Utility User Interface

Limitations of Translation

There are two kinds of translation limitations:

1. Limitations due to the Intellution "Web export."
2. Incompatibility issues between ProcessView and Intellution FIX.

Translator Limitations

This release does not translate the following:

Alarm and Trend viewers

Dynamic properties of FIX Picture Elements that are not supported by the FIX to GraphWorX translator v 6.1.97:

- **EndPoint1 and EndPoint2:** EndPoint dynamic properties are not translated for any line elements.
- **Latched & Current Alarm Colors:** Color Threshold definitions are set with respect to current value only (latched and current alarm settings are not supported).
- **Blink on a new alarm:** Blink on a new alarm dynamic property is not supported for all three color dynamics: Foreground, Edge and Background.
- **Color dynamics:** Even when GraphWorX is capable of changing frame and fill colors at the same time, FIX elements having foreground and edge dynamic properties set are translated as if there were two elements instead: body(fill) and edge. This is because color dynamic of each element can be connected to different data source.

DDE Tags Translation Limitation: DDE Tags used in FIX are divided into three parts, which are aliased and compose OPC tags in GraphWorX:

- DDE tag syntax: nodename.servername.tagname
- OPC aliased tag: \\<<nodename>>\<<servername>>\<<tagname>>

Intellution "Web Export" Limitations (v1.5)

The current release of FIX Web Server does not support all elements that can be included in FIX pictures.

FIX Picture Elements that are NOT Supported by the FIX Web Server export utility

Tag Groups and Variables: Objects that contain tag groups or variables lose any dynamic properties containing these references. The object still appears in the converted picture.

Command Language Scripts: Any command script that is attached to an object is ignored. The object will appear in the converted picture with no script information.

Text Link Background Color: The background color of any text-derived object, including static text, Data Links, Time Links, Date Links, and System Links, is not converted. These objects are displayed as if they had no background color defined.

Scaled Text Link Items: A text-derived object (static text, Data Link, Time Link, Date Link, System Link) that has been scaled in FIX Draw by dragging a control handle will not be converted in its scaled size. It will appear in the converted picture in its original size.

To address this issue, use the following procedure:

1. Open the picture in Fix Draw.
2. Select the text item(s).
3. Use the **Font** menu to select a new font size for the text item. The new size must be different from the default size shown in the dialog.
4. Click **OK** to accept the new size. You may need to repeat this process until the text appears in the correct size in the Draw picture.
5. Save the picture and reconvert using the FIX Web Server Conversion Utility.

Invisible Colors: Objects with "invisible" colors in FIX threshold tables allow other colors to show through. When converted, the threshold tables with invisible colors behave slightly differently. If the threshold color is visible, the object is colored correctly. However, for those threshold values that would create an "invisible" color, the entire object becomes invisible. This is true for any color threshold: foreground, background, edge, or alarm.

Background Color Thresholds: In FIX Draw, background color thresholds work the same way as foreground thresholds; that is, you never really see the background color if the foreground is visible. When converted, the background color of the object becomes visible. In order to duplicate the behavior of FIX View, Intellution recommends changing objects with backgrounds to use foreground thresholds instead.

Fill Styles: Only solid and hollow fill styles are supported by FIX Web Server. Other fill styles are converted to solid fill.

FIX Web Server Conversion Utility

The FIX Web Server Conversion Utility is available as a part of the FIX Web server installation. This utility will convert FIX picture files (.odf) to the Web server files (.jdf, .bmp).

For more information about FIX Web Server Conversion Utility, refer to FIX Web Server Help.

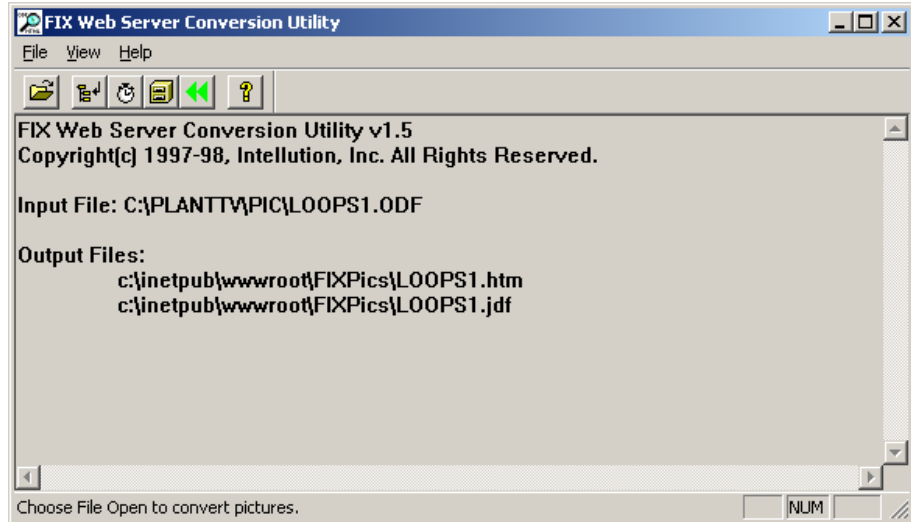


Figure 2. FIX Web Server Conversion Utility

WonderWare to GWX Translator

Introduction

The *InTouch To GraphWorX Translator* uses SAX2 to parse and convert exported .xml InTouch windows into .gdf files. Therefore, implementation of XML3.0 or higher must be installed on your computer in order to run this utility successfully.

Important Note

It is highly suggested that you run *InTouch to GraphWorX Translator* on a machine with *GraphWorX v7.0* (or *ProcessView v7.0*) installed. Since WonderWare pictures are saved in JPG or PNG format and GraphWorX v6.1 does not support importing *.jpg and *.png files, you can get expected results only with GraphWorX v7.0. The translator utility will still work with GraphWorX v6.1 but no picture will be created inside the converted window.

InTouch To GraphWorX Translator

The **InTouch to GraphWorX Translator** was designed to provide the conversion of WonderWare™ InTouch™ windows into GraphWorX displays as easily as possible. It can provide the conversion either for a single window, set of windows or the whole project.

You can specify file(s) to export in the **Select Win-XML Exporter file** edit box (or browse for the file by clicking the **Browse** button). You can either select single window file (*.xml), Window Set file (wwcfg.xml) or Project file (*.wxe) as the source for translation. By default, it points to the Win-XML Exporter project file, which was created or worked on the last time the Win-XML Exporter was launched. (For more information about Win-XML Exporter or its installation instruction, please, refer to the Win-XML Exporter section below.)

Then, fill in the **Destination Directory** the path where you want to place your converted GraphWorX display files. By default, the ProcessView Working Directory is preset in the edit box. The ProcessView working directory can be changed using the GraphWorX application by selecting **Set Working Directory** from the **Tools** menu.

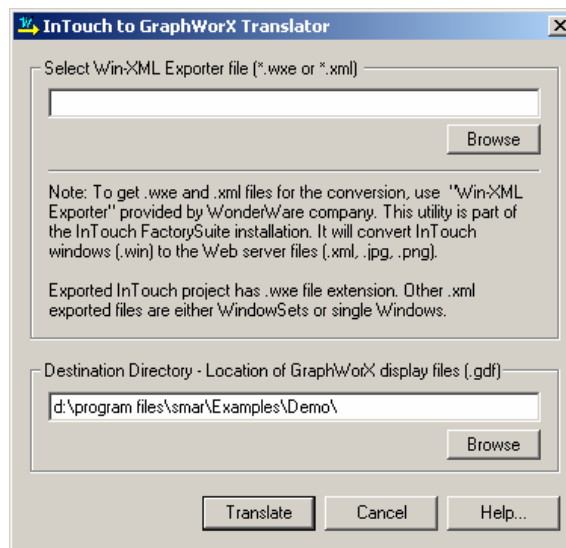


Figure 1. InTouch Translator Utility

Conversion Specifics

The WwToGwxTranslator utility creates one GraphWorX display file per each exported InTouch window file. When the Window Set file is selected, it converts all windows within the Window Set.

When the project file is selected, all Window Sets included in the project are converted. Subdirectory named after the Window Set is created for all project Window Sets. All windows in all Window Sets being converted are created under the corresponding subdirectory.

Supported Conversion Types

Here is the table with conversion pairs – InTouch object on the left is converted into GraphWorX object on the right.

Static Objects

WonderWare InTouch	GraphWorX
Line	GwxLine
H/V Line	GwxLine
Polyline	GwxLine
Polygon	GwxLine
Rectangle	GwxRectangle
Rounded Rectangle	GwxRectangle
Text	GwxText
Bitmap	GwxBitmap
Real-time Trend	NOT CONVERTED
Historical Trend	NOT CONVERTED
Button	GwxButton

Dynamic Objects

WonderWare InTouch		GraphWorX
Line Color	Discrete	GwxDigitalColor
	Analog	GwxDigitalColor
	Discrete Alarm	NOT CONVERTED
	Analog Alarm	NOT CONVERTED
Fill Color	Discrete	GwxDigitalColor
	Analog	GwxDigitalColor
	Discrete Alarm	NOT CONVERTED
	Analog Alarm	NOT CONVERTED
Text Color	Discrete	GwxDigitalColor
	Analog	GwxDigitalColor
	Discrete Alarm	NOT CONVERTED
	Analog Alarm	NOT CONVERTED
Object Size	Height	GwxSize
	Width	GwxSize
Location	Vertical	GwxLocation
	Horizontal	GwxLocation
Percent Fill	Vertical	GwxSize
	Horizontal	GwxSize
Miscellaneous	Visibility	GwxHide
	Blink	GwxFlash
	Orientation	GwxRotation
Value Display	Disable	GwxHide
	Discrete	GwxProcessPoint
	Analog	GwxProcessPoint
	String	GwxProcessPoint
User Inputs	Discrete	NOT CONVERTED
	Analog	NOT CONVERTED
	String	NOT CONVERTED
Sliders	Vertical	GwxLocation
	Horizontal	GwxLocation
Touch Pushbuttons	Discrete Value	GwxPick
	Action	NOT CONVERTED
	Show Window	GwxPick
	Hide Window	NOT CONVERTED

Functional Limitations

Functional limitations of *InTouch to GraphWorX Translator* utility are given by the limitation of the GraphWorX OLE automation interface and limitations of Win-XML Exporter utility.

The *InTouch to GraphWorX Translator* does not translate expressions and does not convert DDE tags into OPC tags.

What Will Not Convert?

The following WindowMaker elements are NOT exported by the Win-XML Exporter:

- **Trends:** The Historical Trend Wizard is not supported in this release of SuiteVoyager Portal.
- **Alarm displays:** SuiteVoyager does not support alarm display wizards. Alarms can be viewed using the Alarms page on the SuiteVoyager Portal.
- **SPC elements:** There is no support for SPC in this release of SuiteVoyager Portal.
- **Scripts:** Window, Condition (except for While True), Key and Data Change scripts are not converted. ActiveX event scripts are converted.

Note: The Win-XML Exporter works only with InTouch 7.1 or greater. Earlier InTouch windows must at minimum be opened and saved in InTouch 7.1 before conversion.

Translation Impurities

Text Object

GraphWorX text object has a gap/space between the string and its frame, while InTouch frame is close-knit to the text string. Therefore, GraphWorX text objects must be resize to approximately reflect the font size of InTouch text object.

Orientation Dynamic

The Orientation dynamic is translated as GwxRotation dynamic. The conversion utility creates the GwxRotation object filled with available attributes. It does not set the pivot and angle of rotation. Adjust them to make them precisely reflect the original situation.

Value Display Dynamics

WonderWare's Value Display dynamic can be directly translated as a process point only if its text object string is empty. When it is not, such Value Display will be translated into two GraphWorX objects, static text object displaying the string and process point displaying the value. The frame of the process point will be placed next to the text object. User then has to adjust its size and position.

Value Display Discrete Dynamic

The GraphWorX equivalent of WonderWare's Value Display Discrete dynamic is a process point with its states set up. Unfortunately, GraphWorX v7.0 automation does not allow setting up the States of a process point. Therefore, the message strings belonging to the discrete values are placed into the custom data field of translated GraphWorX process point.

To finish the translation of Value Display Discrete dynamic, please remove these strings (from custom data field) and use them to configure the states in State Field Configuration dialog. The first line of the custom data is the WonderWare's "On Message," while the second line is "Off Message."

Win-XML Exporter

Installation Instructions

The Win-XML Exporter utility is distributed with Wonderware' SuiteVoyager.

Note: You must have InTouch WindowMaker installed on your computer before installing the Win-XML Exporter.

To install the Win-XML Exporter:

1. From the SuiteVoyager Portal Home Page (obtain the URL or address from your SuiteVoyager Administrator) click the Converter Tool download link. The **File Download** dialog box appears.
2. Choose **Run this program from its current location**. Click **OK**. The **Security Warning** dialog box appears. Click **Yes**. The Suite Voyager Win-XML Exporter Welcome dialog box appears. Click **Finish**.
3. The **Choose Setup Language** dialog box appears. English is the default. Select the language you want and click **OK**. Setup begins and the **Welcome** dialog box appears. Click **Next**.

4. Choose the destination folder. This is where the Win-XML Exporter will be installed. Click **Next**.
5. The files are installed. When the **Setup Complete** dialog box appears, click **Finish**.

For more information about Win-XML Export, refer to its Application Design Guide.



Figure 2. Win-XML Exporter Installation Running

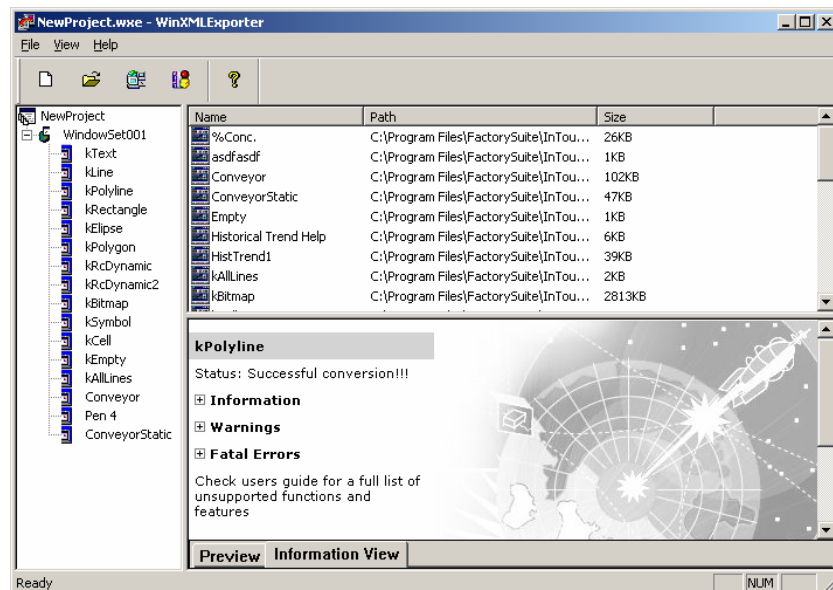


Figure 3. Win-XML Exporter

GraphWorX Viewer ActiveX

Introduction

The GraphWorX Viewer ActiveX ("GWXview32.ocx") is an ActiveX control capable of running GraphWorX displays. The advantage of ActiveX controls like GraphWorX is that they can be embedded into any control container (e.g. Visual Basic Forms, Internet Explorer HTML pages, GraphWorX displays, etc.).

GraphWorX is essentially a runtime-only component (with some minimal configuration capability); the runtime-only design allows the ActiveX to be compact with respect to memory usage. The GraphWorX Viewer ActiveX executes displays created by GraphWorX.exe. GraphWorX has all the runtime capabilities of GraphWorX.exe, except the ability to execute VBA scripts.

Inserting the GraphWorX Viewer ActiveX

Techniques for inserting an ActiveX control may vary slightly among different control containers, however the basics are the same. This section describes how to insert the GraphWorX Viewer ActiveX into GraphWorX.exe.

From the **Edit** menu, choose **Insert New Object**, or click the **Insert ActiveX Control/OLE Object** button on the **ActiveX** toolbar, as shown below.



Figure 1. Insert ActiveX Control/OLE Object Toolbar Button

This opens the **Insert Object** dialog box, shown in the figure below:

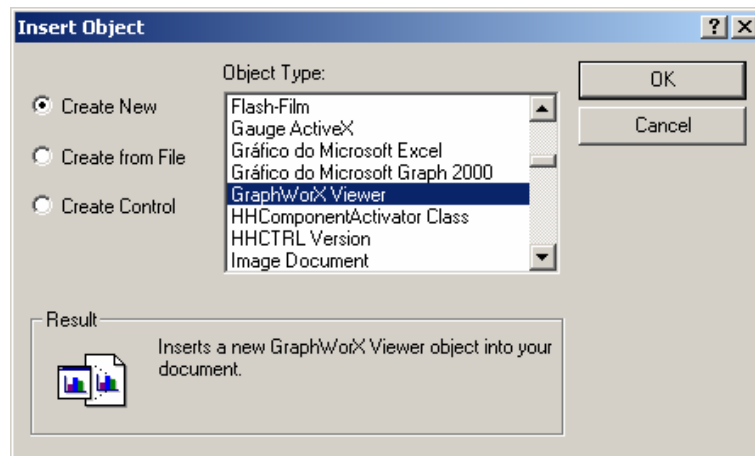


Figure 2. Insert Object Dialog Box

Select **SMAR GWXView ActiveX** from the list of available controls, and then click **OK**. The GraphWorX Viewer ActiveX control will appear in the center of the display. Alternatively, you can click the **GWXView** button on the **ActiveX** toolbar to directly insert the GraphWorX ActiveX.

Configuring the GraphWorX Viewer ActiveX

This property page allows you to change certain attributes of the GraphWorX Viewer ActiveX control.

To configure the GraphWorX Viewer ActiveX:

1. Double-click the GraphWorX Viewer ActiveX to display the control's **Properties** dialog box, shown below.
2. Fill in the parameters described in the **GraphWorX Viewer ActiveX Properties** section below, and then click **OK**.

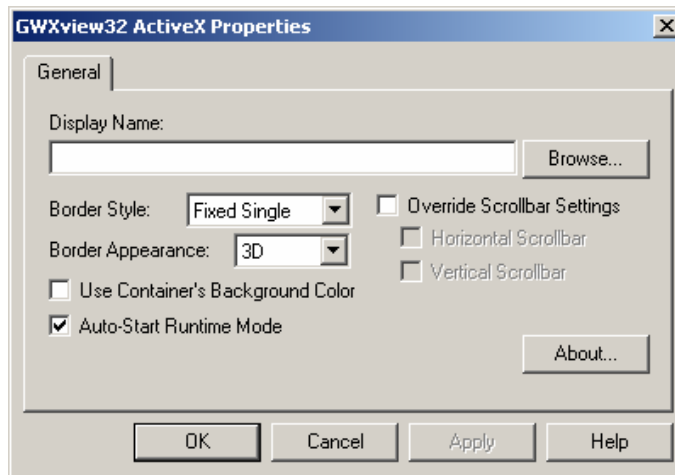


Figure 3. GraphWorX Viewer ActiveX Properties Dialog Box

GraphWorX Viewer ActiveX Parameters

Parameter	Description
Display Name	Specifies the file name of the GraphWorX display to be loaded in this control. Click the Browse button to search for display files.
Border Style	Specifies the appearance of the window border: "No Border" or "Fixed Single."
Border Appearance	Specifies the appearance of the window border: flat or three-dimensional.
Use Container's Background Color	When this option is checked, the background color of the control is automatically set to match the background color of the container in which this control is embedded. This property only works if the container supports the "AmbientBackColor" property. GraphWorX and Visual Basic Forms both support this property.
Auto-Start Runtime Mode	When checked, the control will automatically enter runtime mode when the control's container enters runtime mode. This property only works if the container supports the AmbientUserMode property. GraphWorX and Visual Basic Forms both support this property.
Override Scrollbar Settings	When this option is checked, this property indicates the control will override the scrollbar visibility settings of the display currently loaded in the control. When it is not checked, the control will use the scrollbar settings defined in the currently loaded display.
Horizontal Scrollbar	When this option is checked, the horizontal scrollbar of the control's window is visible. When it is not checked, the horizontal scrollbar is hidden. This property is ignored if Override Scrollbar Settings is not checked.
Vertical Scrollbar	When this option is checked, the vertical scrollbar of the control's window is visible. When it is not checked, the vertical scrollbar is hidden. This property is ignored if Override Scrollbar Settings is not checked.

Note:

