



Your comments and suggestions on the operation of this software are welcome. Please address them to:

## ICONICS

100 Foxborough Blvd.

Foxborough, MA 02035

Tel: 508-543-8600

Fax: 508-543-1503

E-Mail: [support@iconics.com](mailto:support@iconics.com)

Web: [www.iconics.com](http://www.iconics.com)

© ICONICS, Inc. All rights reserved.

## Contents

Introduction to the GraphWorX32 Translation Utility  
Converting Displays  
Translator Controls  
Option Controls  
Script Language Reference

# Introduction to the GraphWorX32 Translation Utility

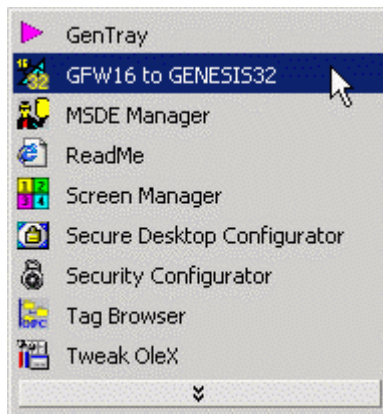
The GraphWorX32 Translation Utility is a 32-bit Windows-based application, that takes a native 16-bit GraphWorX display or any text file using a script language as input and generates a native 32-bit GraphWorX display from it.

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 GraphWorX32 Automation interface functions.

## GraphWorX32 Translation Utility

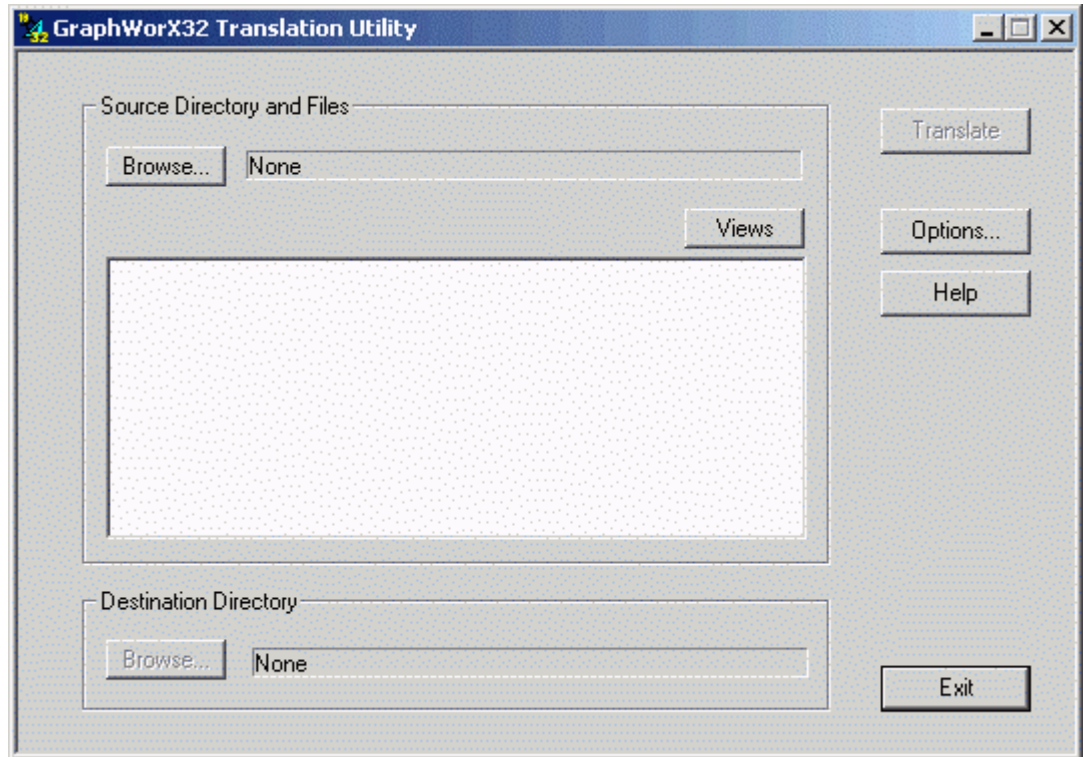
## Converting Displays

Before you start using the translation utility, make sure that you have GraphWorX32 installed. The GENESIS-32 installation provides GraphWorX32. To open the GraphWorX32 Translation Utility from the Windows **Programs** menu, select - **ICONICS GENESIS32- Tools - GFW16 to GENESIS32**, as shown below.



### Starting the GraphWorX32 Translation Utility

This opens the GraphWorX32 Translation Utility screen, shown below.



**GraphWorX32 Translation Utility Screen**

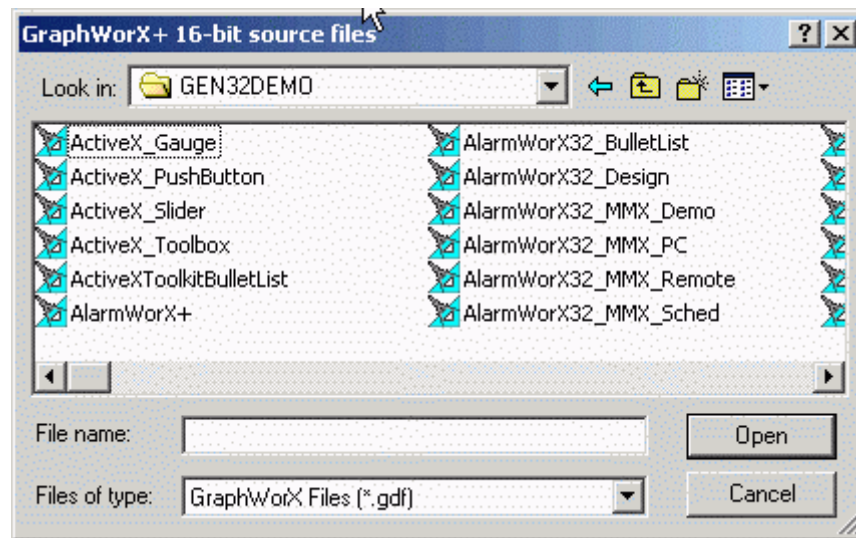
## Native 16-bit GraphWorX displays

This section describes the steps required to translate native 16-bit GraphWorX displays to native 32-bit GraphWorX displays with ICONICS GraphWorX32 Translator.

**Defining Sources**  
**Defining a Destination**  
**Translating**  
**Using Options**

### 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.

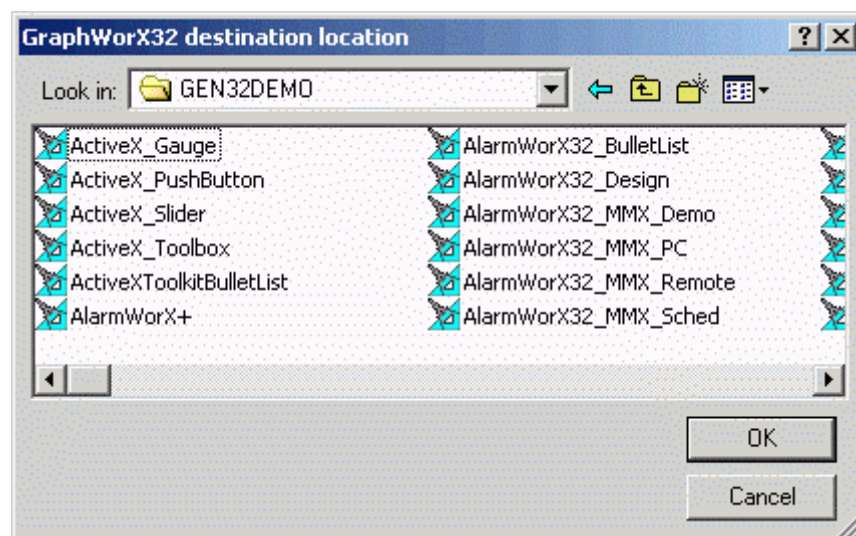


**GraphWorX + 16-Bit Source Files Dialog Box**

## 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 **GraphWorX32 Destination Location** dialog box opens, as shown below. In this dialog box, your selected files are not used. Only the directory name is used. Go into the desired location and click the **OK** button. If accepted, the new destination location is shown in the **Destination Directory** section.



## GraphWorX32 Destination Location Dialog Box

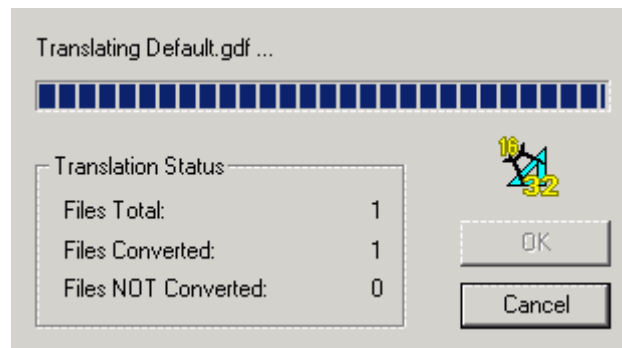
### Translating

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.



### Translation

### Using Options

You can override a few properties in GraphWorX32 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 GraphWorX32 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

### Third-Party Displays

This section describes the steps required to translate third-party displays to native 32-bit GraphWorX displays using ICONICS GraphWorX32 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.

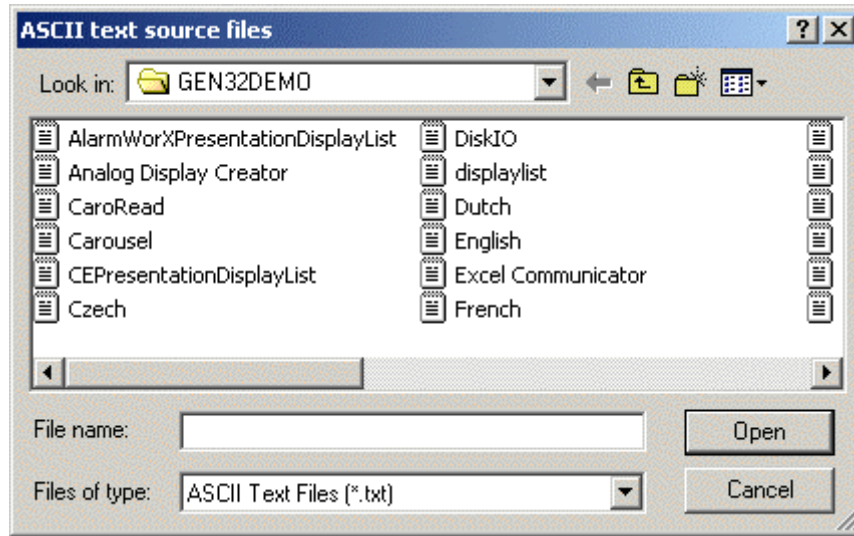
**Defining Sources for a Third Party**  
**Defining a Destination**  
**Translating for Third Party**  
**Using Options for Third Party**

### Converting Displays

### 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.



**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 **GraphWorX32 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.

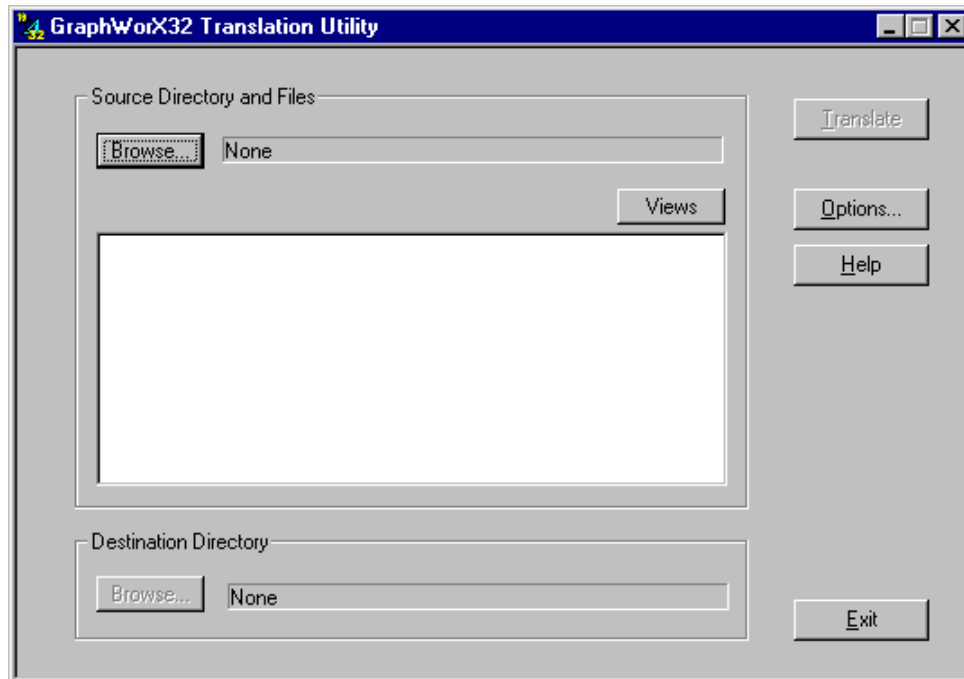
# Translator Controls

This section describes the Translator Controls.

**Main Window**  
**Browsing Source Files**  
**Browsing the Destination Directory**

## Main Window

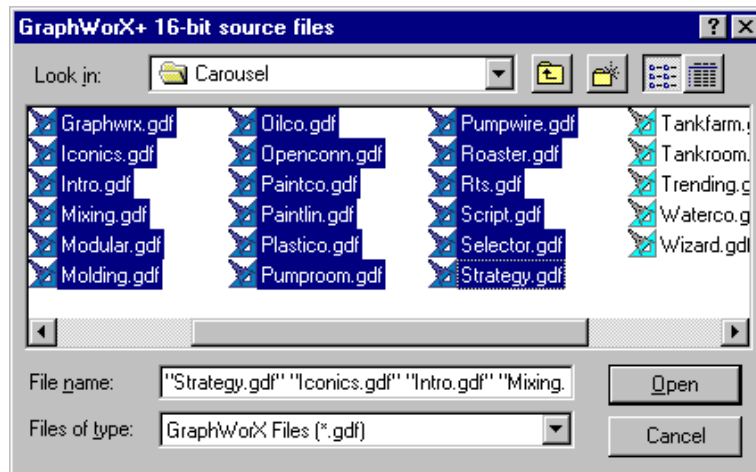
As you launch the application the following main window displays. You use this to operate the translator:



**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:

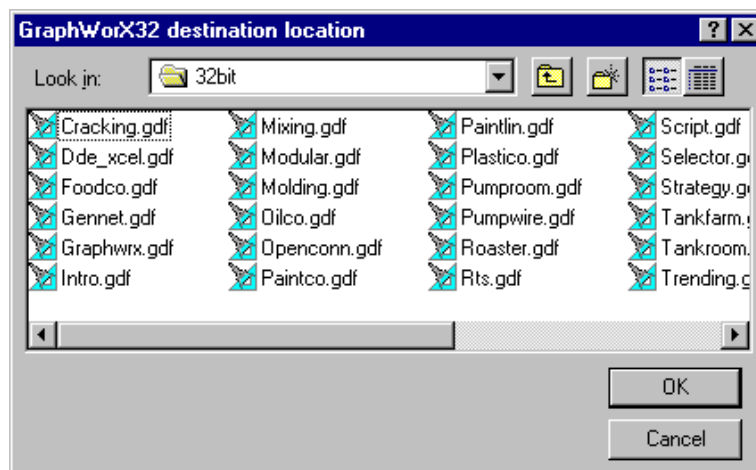


**GraphWorX+ 16-Bit Source files Dialog**

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:



**GraphWorX32 destination location Dialog**

Note that the destination directory must differ from the source. This is to prevent the files from being overwritten.

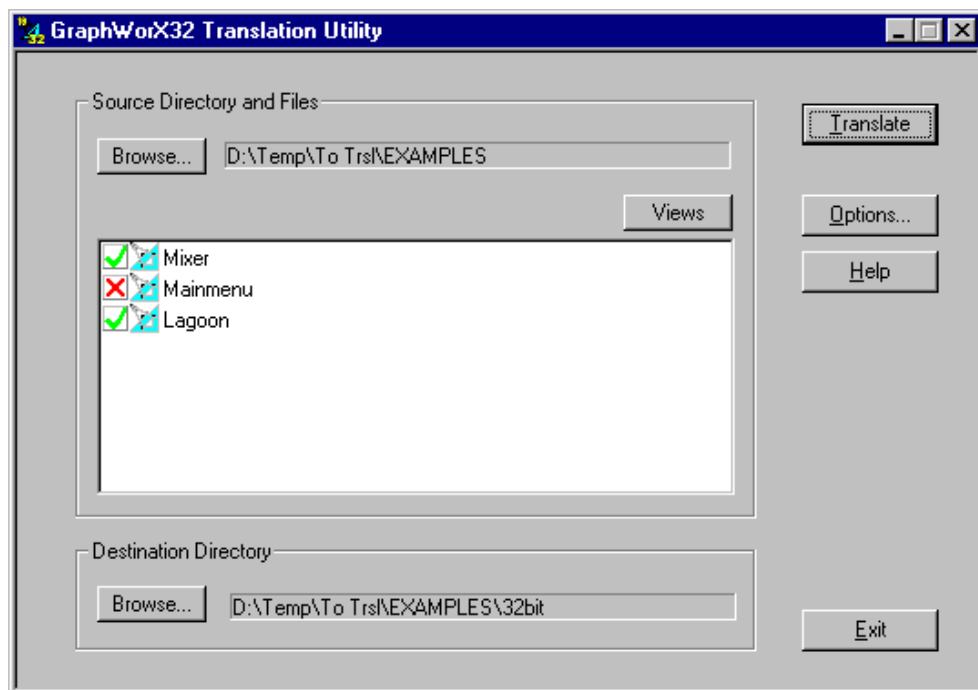
## Source File List Control

The **Views** button provides four different views on the files both translated and not translated. The four modes are:

**List View**  
**Report View**  
**List View and List Box**  
**List View and GWX32 Preview**

### List View

The list view shows all the files you selected for translation, as shown below.

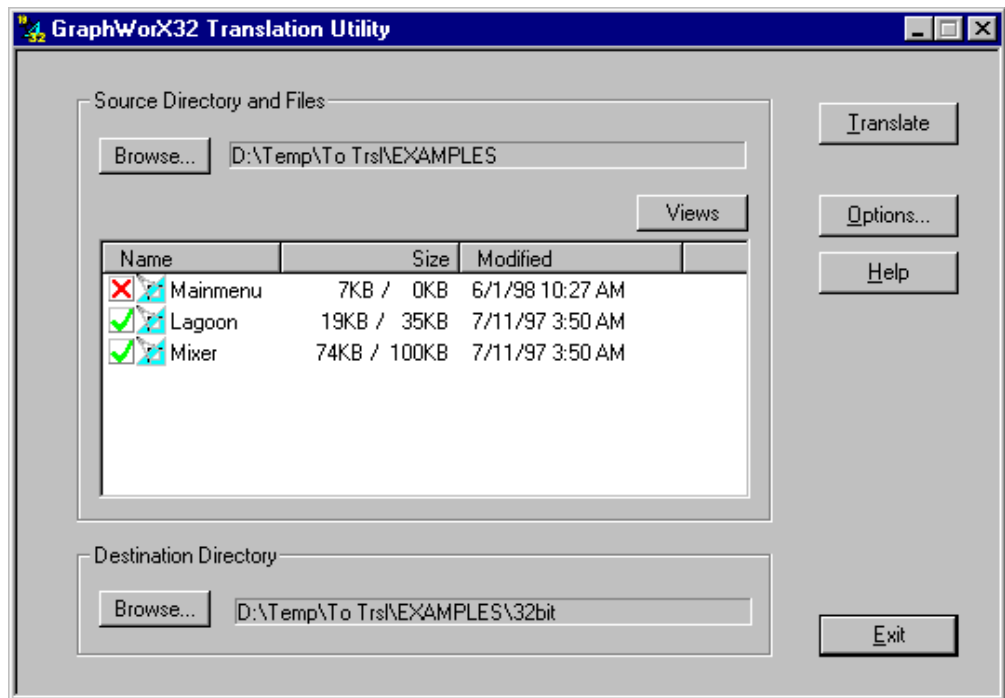


**GraphWorX32 Translation Utility: List View**

## 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.

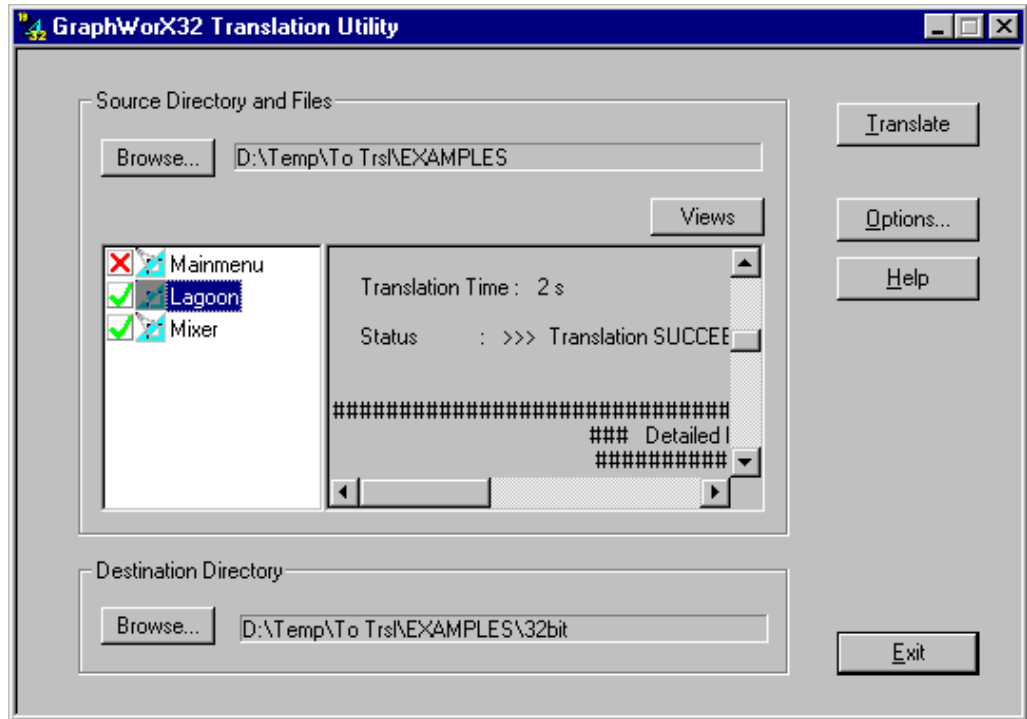


**GraphWorX32 Translation Utility: Report View**

**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.



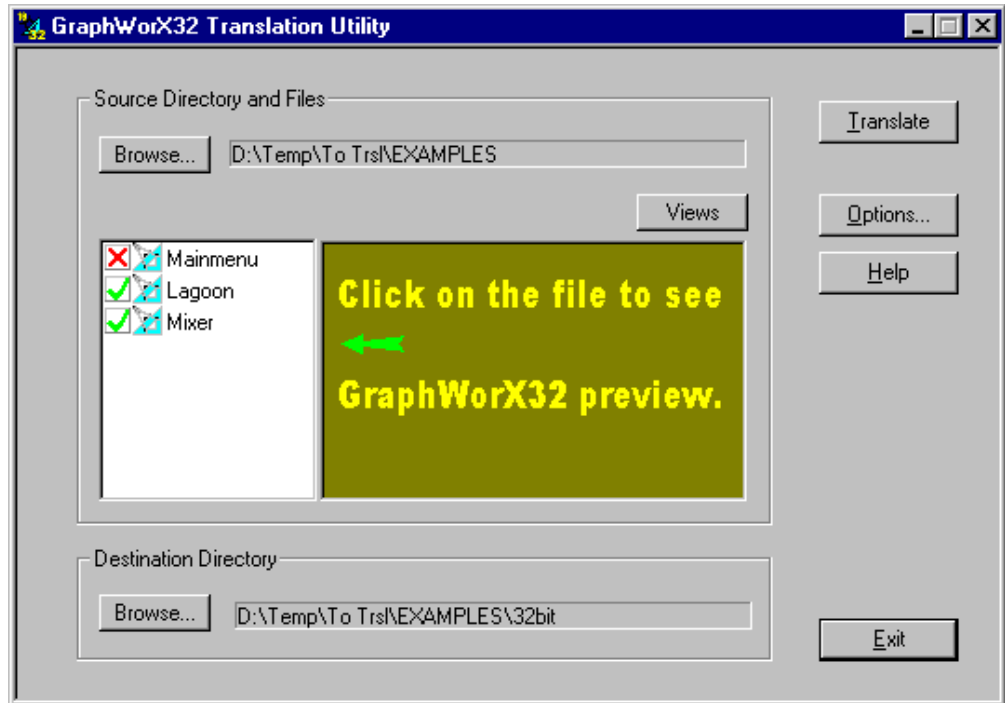
**GraphWorX32 Translation Utility: List View and List Box**

## List View and GraphWorX32 Preview

When you click on one of the translated files, GraphWorX32 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.



**GraphWorX32 Translation Utility: List View and GraphWorX32 Preview**

## Option Controls

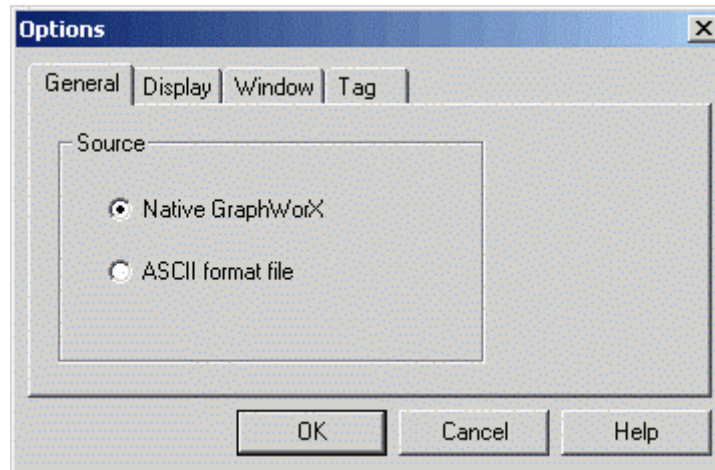
Option controls basically give you a method of overriding displays properties globally, i.e. for all translated displays.

Clicking the **Options** button on the main screen opens the Options dialog box, in which the following options can be modified:

**General**  
**Display**  
**Window**  
**Tag**

### General

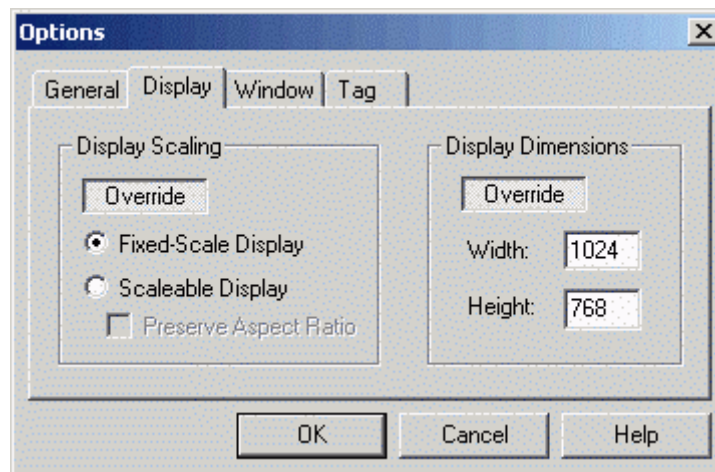
The **Source** section of the **General** tab of the **Options** dialog box, shown below, has two items. When **Native GraphWorX** is selected, the translator will take 16-bit displays as a source. Otherwise, \*.txt files will be taken.



Options Dialog: General Tab

## Display

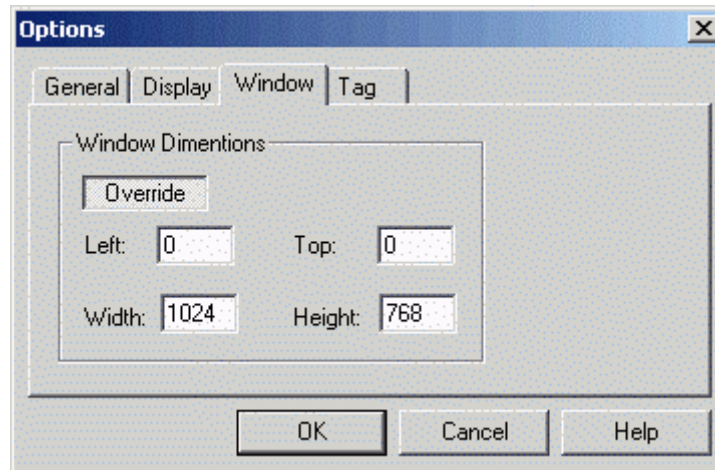
In the **Display** tab of the **Options** dialog box, shown below, you can override **Display Scaling** and **Display Dimensions**. In that case, the setting in the source files will be ignored.



Options Dialog Box: Display Tab

## Window

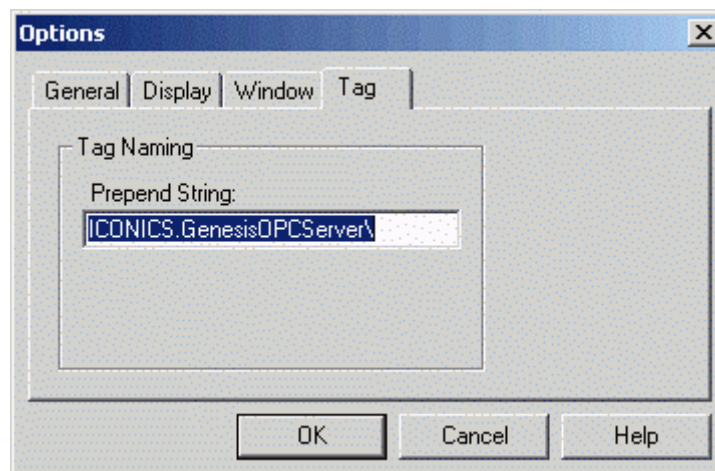
The **Window** tab of the **Options** dialog box, shown below, contains a **Window Dimensions** field, which will set the GraphWorX32 application window position and size.



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.



Options Dialog Box: Tag Tab

## Script Language Reference

This section describes the Script Language Reference used in the GraphWorX32 Translation Utility.

OLE Automation is used to access the properties and methods of objects in GraphWorX32.

Properties are used to reference attributes of an object (for example, a GraphWorX32 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 GraphWorX32 Translation Utility as a response to ASCII text file either provided or generated.

There are numerous object types in GraphWorX32, each with their own methods and properties. This section describes in detail, the objects supported by GraphWorX32 Translation Utility.

Generally, objects in GraphWorX32 can be placed in two groups: **visible** and **dynamic** objects.

**Syntax**  
**Syntax**  
**Keyword Keys**  
**Window Keys**  
**Visible Keys**  
**Button**  
**Dynamics Keys**

## Syntax

Script language is supposed to be used in a pure ASCII text file. Syntax for 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



## Script Language Reference

## 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.**



**Script Language Reference**

## 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 GraphWorX32 main window size and location in pixels.

## **WINDOW\_PERCENT**

**Parameters:** float left, float top, float width, float height

Sets the GraphWorX32 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 GraphWorX32 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 lfCharSet, byte lfOutPrecision, byte lfClipPrecision, byte lfQuality, byte lfPitchAndFamily, BSTR lfFaceName**

### **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, GWX3DEGESTYLE 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, GWX3DEGESTYLE 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, GWX3DEDGESTYLE 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 GraphWorX32 objects (ellipses, rectangles, etc.) are derived. In other words, all visible GraphWorX32 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). GraphWorX32 will ensure that object names are unique. If you assign an object name that already exists for another visible object in a display, GraphWorX32 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 GraphWorX32 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 GraphWorX32 local variables. **Point** objects cannot be explicitly created or destroyed. GraphWorx32 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 GraphWorX32 objects (size dynamic, location dynamic, etc.) are derived. In other words, all dynamic GraphWorX32 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 GraphWorX32 local variable.

### **boolean RangeOverride**

Gets/Sets range override status. When FALSE, GraphWorX32 will use the ranges associated with the primary **DataSource**. When TRUE, GraphWorX32 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 GraphWorX32 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 GraphWorX32 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.